Attack Graph Approach to Dynamic Network Vulnerability Analysis and Countermeasures.

Thaier Hamid

ATTACK GRAPH APPROACH TO DYNAMIC NETWORK VULNERABILITY ANALYSIS AND COUNTERMEASURES

By

Thaier K. A. Hamid

A thesis presented to the University of Bedfordshire

In partial fulfilment of the prerequisites for the degree of Doctor of Philosophy

University of Bedfordshire

ATTACK GRAPH APPROACH TO DYNAMIC NETWORK VULNERABILITY ANALYSIS AND COUNTERMEASURES

By

Thaier K. A. Hamid

ABSTRACT

It is widely accepted that modern computer networks (often presented as a heterogeneous collection of functioning organisations, applications, software, and hardware) contain vulnerabilities. This research proposes a new methodology to compute a dynamic severity cost for each state. Here a state refers to the behaviour of a system during an attack; an example of a state is where an attacker could influence the information on an application to alter the credentials. This is performed by utilising a modified variant of the Common Vulnerability Scoring System (CVSS), referred to as a Dynamic Vulnerability Scoring System (DVSS). This calculates scores of intrinsic, time-based, and ecological metrics by combining related sub-scores and modelling the problem's parameters into a mathematical framework to develop a unique severity cost.

The individual static nature of CVSS affects the scoring value, so the author has adapted a novel model to produce a DVSS metric that is more precise and efficient. In this approach, different parameters are used to compute the final scores determined from a number of parameters including network architecture, device setting, and the impact of vulnerability interactions.

An attack graph (AG) is a security model representing the chains of vulnerability exploits in a network. A number of researchers have acknowledged the attack graph visual complexity and a lack of in-depth understanding. Current attack graph tools are constrained to only limited attributes or even rely on hand-generated input. The automatic formation of vulnerability information has been troublesome and vulnerability descriptions are frequently created by hand, or based on limited data. The network architectures and configurations along with the interactions between the

individual vulnerabilities are considered in the method of computing the Cost using the DVSS and a dynamic cost-centric framework.

A new methodology was built up to present an attack graph with a dynamic cost metric based on DVSS and also a novel methodology to estimate and represent the cost-centric approach for each host' states was followed out.

A framework is carried out on a test network, using the Nessus scanner to detect known vulnerabilities, implement these results and to build and represent the dynamic cost centric attack graph using ranking algorithms (in a standardised fashion to Mehta et al. 2006 and Kijsanayothin, 2010). However, instead of using vulnerabilities for each host, a CostRank Markov Model has developed utilising a novel cost-centric approach, thereby reducing the complexity in the attack graph and reducing the problem of visibility.

An analogous parallel algorithm is developed to implement CostRank. The reason for developing a parallel CostRank Algorithm is to expedite the states ranking calculations for the increasing number of hosts and/or vulnerabilities. In the same way, the author intends to secure large scale networks that require fast and reliable computing to calculate the ranking of enormous graphs with thousands of vertices (states) and millions of arcs (representing an action to move from one state to another). In this proposed approach, the focus on a parallel CostRank computational architecture to appraise the enhancement in CostRank calculations and scalability of of the algorithm. In particular, a partitioning of input data, graph files and ranking vectors with a load balancing technique can enhance the performance and scalability of CostRank computations in parallel.

A practical model of analogous CostRank parallel calculation is undertaken, resulting in a substantial decrease in calculations communication levels and in iteration time. The results are presented in an analytical approach in terms of scalability, efficiency, memory usage, speed up and input/output rates.

Finally, a countermeasures model is developed to protect against network attacks by using a Dynamic Countermeasures Attack Tree (DCAT). The following scheme is used to build DCAT tree (i) using scalable parallel CostRank Algorithm to determine the critical asset, that system administrators need to protect; (ii) Track the Nessus

scanner to determine the vulnerabilities associated with the asset using the dynamic cost centric framework and DVSS; (iii) Check out all published mitigations for all vulnerabilities. (iv) Assess how well the security solution mitigates those risks; (v) Assess DCAT algorithm in terms of effective security cost, probability and cost/benefit analysis to reduce the total impact of a specific vulnerability.

ACKNOWLEDGEMENTS

I would like to thank my supervisors' team, special thanks to my Director of Study Paul Sant for guiding me in the last stage to completion of this thesis.
Extra thanks to Professor Della Freeth, for her kind helps and to the oral (viva) examination board members and to the academic and technical staff in LSC for giving me a valuable feedbacks regarding the Business case study and the science lab setting for parallel CostRank algorithm experiments.
Lastly, I would wish to pledge this work to my son "Khalid," who added up to this world on 07/07/2014, and to my family and to the soul of my parents.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

BSP	Bulk-Synchronous parallel model
CVSS	Common Vulnerability Scoring System
CVE	Common Vulnerabilities and Exposures
NVD	National Vulnerability Database
AT	Attack Trees
AG	Attack Graphs
DCAT	Dynamic Countermeasures Attack Trees
DCCAG	Dynamic Cost Centric Attack Graph
DVSS	Dynamic Vulnerability Scoring System
E_P	Efficiency of parallel CostRank algorithm
EUT	Expected Utility Theory
IRR	Internal Rate of Return
METF	Mean Effort-To-Failure
Net SPA	Network Security and Planning Architectures
OLC	Operational Level Cost
PEN	Penetration Testing
P	Probability
P_S	Problem size
SP	Speedup
SWOT	Strength, Weaknesses, Opportunities, and Threats
SMART	Specific, Measurable, Agreed upon, Realistic, Time-bound
T_O	Overhead Time
$T_{Parallel}$	Parallel Runtime
$T_{sequential}$	Sequential Runtime

DECLARATION

I declare that:

1. This thesis is my own unaided work. It is being submitted for the degree of Doctor of Philosophy at the University of Bedfordshire.
2. The following, fully acknowledged, figures (include all diagrams, charts and photographs, etc.) were taken from published works. I understand that these diagrams will be blanked out in the electronic version of my thesis that is stored in the University repository.
3. No material contained in the thesis has been used in any other submission for an academic award.
4. Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
5. Either none of this work has been published before submitting, or parts of this work have been published as:

- Hamid, Thaier, and Carsten Maple. "A Graph theoretical approach to Network Vulnerability Analysis and Countermeasures." (2011).
- Hamid, Thaier, Carsten Maple, and Paul Sant. "Methodologies to develop quantitative risk evaluation metrics." (2012).
- Hamid, Thaier, Carsten Maple, and Yong Yue. "A parallel algorithm to calculate the CostRank of a network." (2012).
- Hamid, Thaier, Carsten Maple. Article: Network Attacks and Countermeasures, International Conference on Education and Information Management (ICEIM-2013), Penang, Malaysia.
- Hamid, Thaier, Carsten Maple. Article: Theoretical And Practical Approaches to Develop a Parallel Cost Rank Algorithm of a Network, Symposium on Emerging Trends in Advanced Computing (SETAC-2012) - 15th May 2012.

Signed:
Date:

CHAPTER 1. INTRODUCTION

1.1 AIMS AND OBJECTIVES OF THE RESEARCH

It is widely accepted that modern computer networks (often presented as a heterogeneous collection of functioning organisations, applications, software, and hardware) contain vulnerabilities. This research proposes a new methodology to compute a dynamic severity cost for each state. Here a state refers to the behaviour of a system during an attack; an example of a state is where an attacker could influence the information on an application to alter the credentials. Through utilising a modified version of the Common Vulnerability Scoring System (CVSS), referred to as a Dynamic Vulnerability Scoring System (DVSS). This calculates scores of intrinsic, time-based, and ecological metrics by combining related sub-scores and modelling the problem's parameters into a mathematical framework to produce a unique severity cost.

The individual static nature of CVSS affects the scoring value, so a novel model has adapted to produce a DVSS metric that is more accurate and effective.

In this approach, different parameters are used to compute the final scores determined from a number of parameters including network architecture, device setting, and the impact of vulnerability interactions.

The main aim of this research is:

To secure Network systems from intended attackers, by providing security professionals and system administrators with techniques and tools to identify vulnerabilities and their impact, in order to support security decisions and to create and maintains security policies and procedures.

The objectives of the research are:

1. To develop a unique severity cost of all vulnerabilities existing in a network host, using dynamic risk impact metric methods, which can improve the risk scoring method within the field of security engineering in the future.

2. To develop a cost-centric approach based on an attack graph model for dynamic cost metric calculations that can introduce practical, useful techniques for detecting and analysing vulnerabilities to secure network systems.
3. To develop a dynamic CostRank methodology to mathematically addresses the optimum solution of ranking attack graphs towards minimising the dimension and complication of attack graphs to assist security professionals and system administrators to examine security decisions and to create and maintain security policies and procedures.
4. To develop an analogous parallel algorithm to make CostRank calculations for an increasing number of hosts in the network in an efficient and scalable manner.
5. To simulate Countermeasure Attack Tree algorithms to calculate the flow of the costs and mitigations across the tree using a dynamic impact approach to find the most effective solution to implement the countermeasures.
6. To conduct extensive testing and experimentation to allocate a solution set.

1.2 PROBLEM STATEMENT

Despite continuous efforts to secure cyber liberty from intentional attackers' missions, control operations certainly experience security incidents, which result in damage to the confidentiality, integrity, or availability (CIA) and ultimately cause the degradation of a system, or even make it unusable.

As an example [Anderson, Ross, et al., 2012] state, the estimated annual cost of cyberspace and network attack is $27bn. To put this into context, it is estimated that more than $1,2bn can be attributed to fraudulent activity within the DWP (Department of Work and Pensions), and more than $3bn lost in income tax fraud while they were in the process of moving all the claims online in 2013, which could make the figures much higher.

The proposed attack graphs are presented as practical tools for detecting and analysing vulnerabilities to secure network systems. Even after forming the attack graphs professionally, often the dimension and complexity of attack graphs deter

humans from completely understanding the information presented. A cleansing of a huge volume of information is vital to assist network administrators in efficiently detecting and analysing the vulnerabilities and in resources earmarked to implement proper mitigation strategies. This thesis presents a novel CostRank algorithm that is an improvement of the PageRank algorithm used by Google to rank web graphs in terms of web pages. The calculations of PageRank and CostRank are based on the stochastic Markov model [Kijsanayothin, 2010], as the computation also depends on finding a fixed probability distribution for a Markov chain in which the states are represented as vertices or nodes.

The CostRank algorithm to be considered a state is important if other important states are linked to it. However, not all states are equal in importance (dynamic cost); the link from different states holds a different cost value. The CostRank algorithm uses the dependency attack graphs and integrates the vulnerability data to measure a severity cost for each state using the Dynamic Vulnerability Scoring System (DVSS) developed based on CVSS using Intrinsic, Time-based and Ecological metrics. Related sub-scores are combined to produce a unique severity cost by modelling the problem's parameters in a mathematical framework to calculate the severity cost for each vertex in the graph in terms of privilege, and cost metrics, which indicate their severity during a multi-step attack against the network. The framework is implemented on different sized networks, using the Nessus scanner to discover known vulnerabilities, and to implement the results to build and represent the cost centric attack graph. Instead of using vulnerabilities for each host, a CostRank Markov Model has developed that reduces the complexity in the attack graph using cost-centric, OLC approaches. The consequence of implementing the parallel CostRank algorithm for a number of different network scenarios demonstrates that the numerical ranking calculations are reliable and consistent with the native severity that the privileges and vulnerabilities have during an attack. The vertex ranking is used to provide resources and prioritises countermeasures using a Dynamic Countermeasure Attack Tree (DCAT), to help better comprehending of the security problems and provide a solution to the visual complexity of attack graphs.

1.3 HYPOTHESES

1- Dynamic risk impact metric methods can improve the risk scoring method, in order to reduce the visual complexity of an attack graph.
2- A Dynamic CostRank methodology can provide the effective solution of ranking attack graphs using a cost-centric approach towards minimising the dimension and complexity of attack graphs.
3- The parallel CostRank algorithm will make the rank calculations for an increasing number of hosts in the networks in an efficient and scalable manner.
4- Countermeasures Attack Tree algorithms can represent the effective solution to implement effective countermeasures and mitigations.

1.4 BACKGROUND AND RELATED STUDIES

Network systems are certainly at risk as they offer services from different machines. They depend on application software providers, which are subject to fault, making them susceptible to malicious attacks. The Internet, which has brought many benefits to organisations and individuals, has also increased the risks of having hosts compromised without the need for physical access. Network and application vulnerabilities refer to potential problems in configurations such as ports, IPs, services or the software construction (programming), installation or configuration errors to offer services such as Transparent Network Substrate (TNS) Listener on Oracle software for database servers, buffer overflow on Windows XP SP2, Vista and 7, to name but a few important examples. This is reflected in the number of published vulnerabilities reported by Secunia [Secunia Report, 2013] and the number of Common Vulnerabilities and Exposures (CVEs) disclosed per year since 2005 with a breakdown of the solution status ("unpatched," "patched," "total number of vulnerabilities"). As shown in Figure 1-1, on average, there were 5,850 CVEs per year from 2005 to 2013. An extrapolation of the data of 2010 showed the numbers of CVEs radically increased, as the

exploits of vulnerabilities in common applications such as Java and Microsoft applications are discovered using security applications (such as IDS, Microsoft security products). It should be noted that older vulnerabilities are more likely to have a patch available than recently found vulnerabilities.

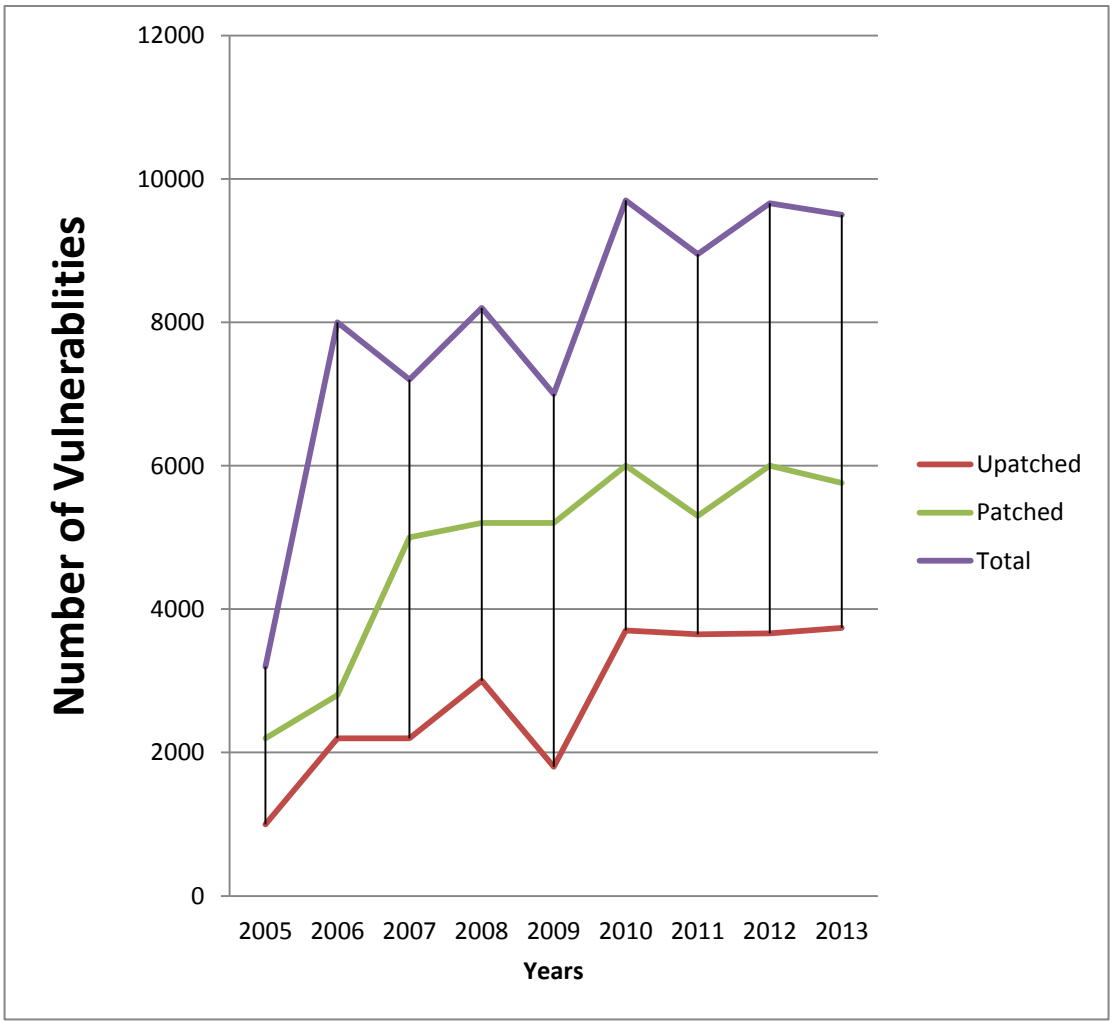


Figure 1-1. Number of published vulnerabilities from 2005-2013, data from [Secunia Report, 2013]

You can see that from 2005 to 2013, a sharp rise in the number of vulnerabilities. In 2006-2007 and from 2010-2011, there was an insignificant decrease of 1.8%. This was due in part to the release of new operating systems at that time.

Nevertheless, in 2008 alone, a total of 8,000 vulnerabilities were published, as shown in Figure 1-1. Consequently, this volume of vulnerabilities requires management from organisations to determine which of the daily reported vulnerabilities apply to their situation according to the software they have installed, their version, and configuration.

At first glance, this appears to be a manageable problem in network administration, because contracted companies can provide the filtering of vulnerabilities, which apply to a specific organisation on a daily basis as a service. Table-1-1 shows the figure of vulnerabilities, including a breakdown by Microsoft operating systems and third party programs [Mellado et al., 2006]**;** as you can notice from vulnerabilities breakdown, the third party programs followed by Microsoft programs installed by the users have the highest number of vulnerabilities. However, a number of factors may turn this into a rather challenging management problem. Firstly, there is a time interval between vulnerability discovery and patch releases [Power et al., 2010]. Network administrators are usually very slow in applying fixes [Prasad, Major Santosh, 2008].

Vulnerabilities – Breakdown						
	2007	2008	2009	2010	2011	2012
Windows XP	39	55	72	47	100	48
Windows Vista	25	49	58	39	91	49
Win 7	-	5	15	34	40	51
Microsoft programs	79	89	85	62	73	77
3rd party programs	120	207	286	275	1024	792

Table 1-1. The number of vulnerabilities by breakdown of Microsoft operating systems, and 3rd party programs [Mellado et al., 2006]

Secondly, even in well-managed networks with rigid security policies, access points to attackers may be totally unidentified by network administrators.

Finally, the vulnerabilities are not exploited in isolation; once an attacker has found a way into a network, they will try to maximize their return on investment. Figure 1-2 shows the relationship between attackers and defenders.

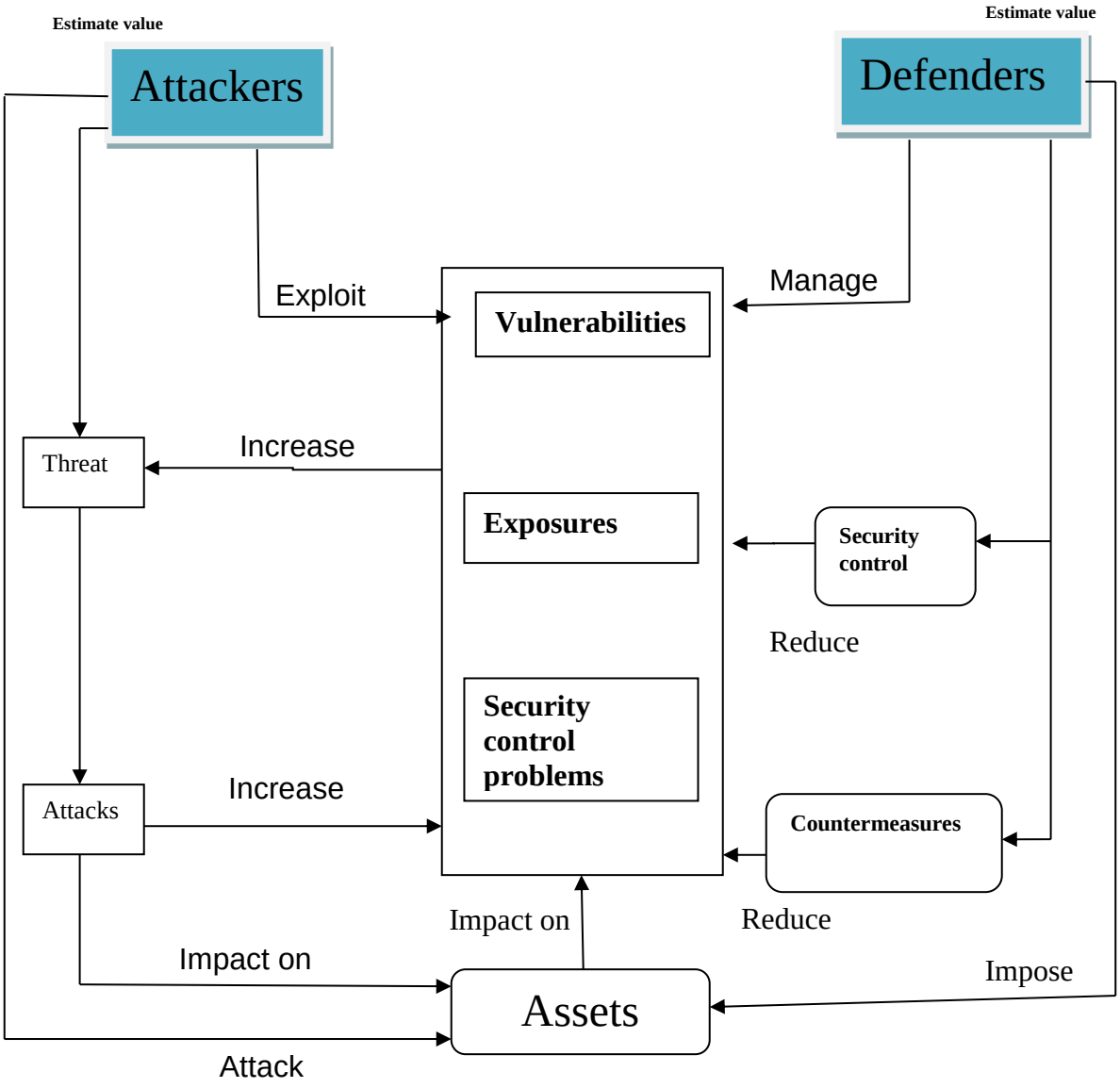


Figure 1-2. The relationship between attackers and defenders

Beneath, an indication is offered about how the problem of finding potential multi-step network attacks has been approached in literature. There are three main streams of work, directly related to this topic: Penetration Testing, Attack Trees, and Attack Graphs.

1.4.1 Penetration Testing

Penetration testing, also known as PEN is an important phase to secure any computer network, system or product; several organisations consider using vulnerability scanners, intrusion detection systems (IDS) and/or other security tools as penetration testing methods. The real PEN should be considered in the stages of design and implementation of a system not only in the operational stage. Mostly, PEN proves the falsity of the hypothesis that a specific network, system or software application is secure.

PEN is a method to check the security strength [He et al., 2006] of a target under evaluation, either for verifying compliance of regulations or for quality assurance purposes. Authorised professionals perform this, reproducing a threat agent, employing the same set of tools and strategies, to break security controls in place. The targets of evaluation can be an organisation's network or application software, or partitions on it, and in this case, the goal usually is to assess the network, applications against the risk of gaining access to sensitive information.

The attack graph represents the chains of vulnerability exploits (obtained from vulnerability scanners). That provide the security professionals and system administrators information regarding possible attack paths that the attackers can use to exploit different vulnerabilities existing in the network hosts to assist them in making security decisions (including implementing security patches and mitigations for different vulnerabilities) and to create and maintain security policies and procedures to preventing the networks from possible potential attacks. Considering different types of attack graphs and vulnerability scanners to discover the weakness in the network system, a dynamic approach to construct DCCAG can be considered as a tool to perform PEN tests.

PEN tests can be conducted in several manners: they could be executed on a white or black box manner. A black box type of test is more appropriate to simulate attacks from outsiders of an organisation. A white box test is while the organisation's staffs have full knowledge about the testing. A PEN test has 4 stages and is usually supported by the Flaw Hypothesis Methodology (FHM) [Prasad, Major Santosh, 2008], as shown in Figure 1-3.

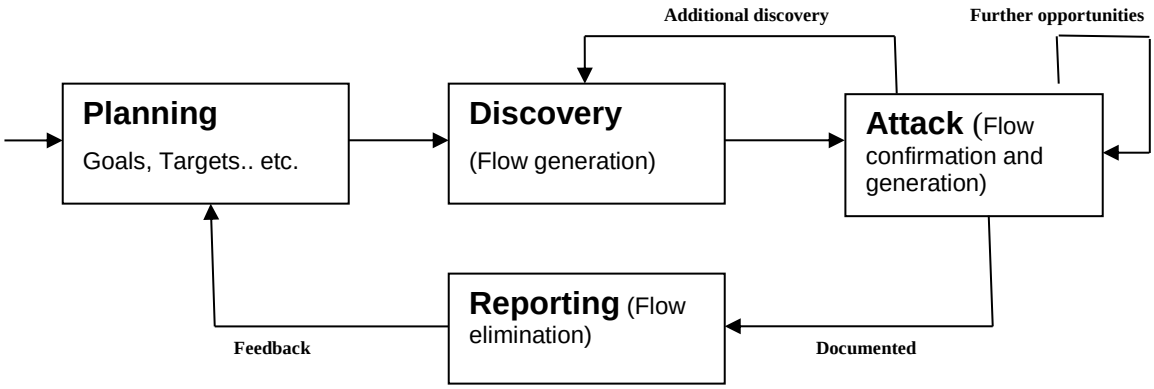


Figure 1-3. Penetration test cycle adapted from [Prasad, Major Santosh, 2008]

[Linde, 1975] stated that PEN used either flaw hypothesis or attack tree, the elementary flaw hypothesis approach as follows:

1. Outline the goals of penetration.
2. Implement system background review.
3. Create hypothetical weaknesses.
4. Approve hypothesis.
5. Specify revealed weaknesses.
6. Eradicate revealed weaknesses.

The attack tree approach, similar to that of a fault tree, is proposed to perform PEN where there is not enough information relating to the system background, in this case the hypotheses will be arranged in a tree structure with OR and AND to approve the hypothesis.

1.4.2 Attack Trees

An Attack Tree (AT) is a multiple level diagram containing one root, multiple leaves, and multiple children. The AT has been used in many of the real world security implementations, such as digital content security protocols and Mobile devices, WAP assisted protocols [Kizza, Joseph Migga, 2013].

There are different types of AT based on the metrics used. For example, vulnerabilities trees, which use vulnerability metrics, defence trees and threat trees.

The root in an AT represents the terminal destination of an attacker, the source node is then refined into sub-goals and the sub-goals are further refined until the leaf node is reached which represents the specific attacker's actions.

In the attack tree, there are two kinds of refining:

1-Conjunction (AND): refined sub-goals are satisfied if all children are true.

2-Disjunction (OR): refined sub-goals are satisfied if any of the children is true.

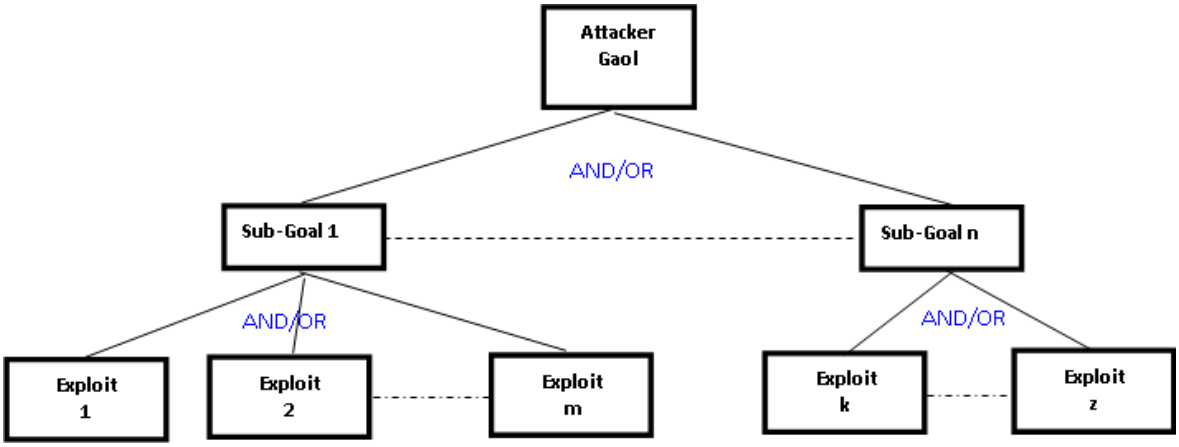


Figure 1-4. Attack tree structure.

Figure 1-4 shows the structure of a classical AT, which demonstrates the following:

1. The root of the tree at first level represents the valued assets to be protected.
2. The intermediate level represents states as security properties that enable the attacker to reach the goal or the next level of the attack.

3. The leaf of the tree, at the third level, represents a different attacker exploit. Each leaf node represents a multi-step attack to move to the next level of attack. ATs use AND & OR refinements, AND as in the general AT [Edge, Kenneth S., 2007] means, the attacker should compromise all leaf node vulnerabilities, to reach the goal or to progress to the next level of attack. While OR refinement means at least one vulnerability should be compromised to reach the goal or to progress to the next level of attack.

Attack Trees are in fact a variation of Fault Trees (FT) applied to the domain of Information Security, see Bruce Schneier [Schneier, 1999]. The root of an FT represents a failure, i.e. an undesired event. The leaves represent causes, which contribute to the parent fails, i.e. basic observable failures. The construction of both types of trees (AT and FT) requires deductive reasoning, which means thinking retrospectively and looking for means or causes of a phenomenon to be avoided. The AT are used in different approaches in order to find a unique severity cost of all vulnerabilities existing in a network host, using dynamic risk impact metrics and in the method of finding the optimum solution to implement the countermeasures and mitigations using CAT.

There are many drawbacks to AT as you will see in the next chapter and therefore the research moves toward the use of Attack Graphs (AG) which are discussed in the next section.

1.4.3 Attack Graphs

A security model representing the chains of vulnerability exploits in a network can be in various forms. The attack graph could be organised [Ou et al., 2005] as a state oriented attack graph, exploit oriented attack graph or a state exploits oriented attack graph. In a state oriented attack graph, the vertices represent a group of the network states and the attacks/exploits represent the edges to show the transitions from one state to a different state in the network.

Early approaches relied on state enumeration (i.e. state of the attacker and the network) to generate Attack Graphs. Model checker graphs are an example of such an approach [Sheyner, Oleg, and Jeannette Wing, 2004], from Carnegie

Mellon University, use a symbolic model checker (NuSMV) to build Attack Graphs. The approach requires the input of a group of machines of a specific network in a fixed condition, representation chosen for use in this model checker [Sheyner, Oleg, et al, 2002]. The model checker builds the model using information related to connectivity between hosts and a library of attack actions specifying details about vulnerabilities in terms of intruder/network preconditions. Model checker approaches in general tend to suffer from the drawback of the state explosion problem [J. Cullum et al., 2007]. This means the complexity of the graph generated, grows exponentially O (2^n), in terms of the size of the state space *n*, in this case *n* represents the number of hosts and vulnerabilities. Therefore, state enumeration-based attack graphs [Borodin, Allan, et al, 2005] [Ding, Chris, et al., 2002] [Mehta, Vaibhav, et al., 2006], also called full graphs, do not scale to real networks. An important aspect of these graphs is the presence of state repetition since states are enumerated in every possible order. In practice, it means that the attacker would visit the same state more than once to reacquire capabilities. Dacier et al. [Steven, Noel, and Sushil Jajodia, 2005] put forward the concept of privilege graph, as the vertices could represent the privileges of an attack originator or might represent a probable target, i.e. nodes (vertices) signify a set of privileges possessed by users or a group of users. Arcs are the vulnerabilities representation, which might permit the gaining of the privileges of a probable target of an attack originator, the privilege graph is then used to construct the state graph (please refer to chapter 3 in this thesis for further details of different types of graphs), if the transition occurs. The author converts a privilege graph to a Stochastic Petri Net obtaining a state graph to eliminate initial duplication of states. However, to estimate the probabilistic Mean Effort To Failure, METF, in other words mean effort for an attacker to exploit a target node, they have to enumerate all possible paths to this target. Experiments with a system containing 13 vulnerabilities, reported in [Ou, Xinming et al., 2006], failed to successfully compute “due to the complexity of the algorithm”, showing that the METF “can only be computed when the number of paths between the attacker and the target is relatively small".

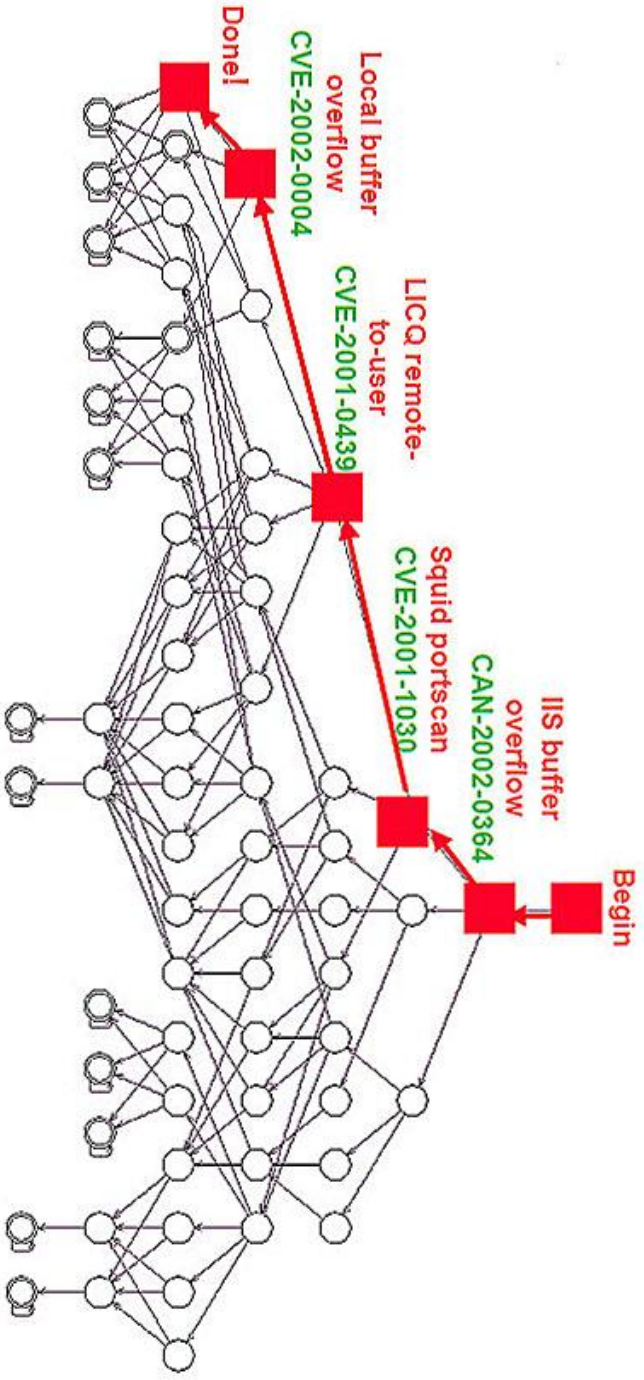


Figure 1-5. Exploit dependency attack graph representations [Idika, 2010]

The 2nd generation of attack graphs (as shown in Figure 1-5) assume that attacks are in the domain of network attacks, as introduced by [Ammann et al., 2002]. It means that the attacker never has to backtrack, i.e. the attacker does not loose capabilities acquired previously in the course of an attack.

In the exploit oriented attack graph, the vertices signify conditions and the edges represent the exploits. This is the converse of a state oriented attack graph and sometimes refers to as an exploit dependency, attack graph. The representation of exploiting dependency/orienting attack graphs comes with primary condition/s and goal condition/s and the states of some unique hosts. The exploit-oriented attack graph's initial state(s) and the goal state(s) of the network are special nodes. Primary condition/s symbolises the exploit hosts with exact post conditions and void/null preconditions. Goal condition/s represents the exploit hosts with exact preconditions and void/null post conditions.
The states and exploits are represented as vertices in the state-exploit oriented attack graph [Jürgenson, Aivo, 2010]. An edge in this graph could relate to a specific state, to a specific exploit or a specific exploit and a state. The edge will not directly relate a state to another state or an exploit to another exploit.
In this thesis, a state oriented attack graph is used. As a state is a network attribute or set of network attributes, network state, such as network nodes, node connectivity, and installed /upgraded/modified software application, which are affected because of attack on a specific node, privilege access right or any other characteristic such as privilege level. The state-oriented attack graph model has been modified to represent the cost metric approach methodology.
The Network Security and Planning Architectures (Net SPA) can represent a large network attack graph in an efficient manner for the purpose of network security analysis; Net SPA collects data from different resources, such as the configuration files of routers and firewalls, vulnerability scanners (Nessus scanner) and vulnerability databases to build and generate the attack graphs. Net SPA, according to existing results of security analysis, produces actions,

recommendations, and suggestions to enhance the network security. However, Net SPA fails to afford efficient manners to represent the enormous amount of information and vulnerabilities [O'Hare, Scott et al., 2008] [Ou, Xinming et al., 2006].

1.4.4 Visual Attack Graphs Complexity

A number of researchers have documented attack graph visual complexity, e.g. [Franqueira, Virginia et al., 2009] [Li, Zhi-tang, et al., 2007] [Williams, Leevar et al., 2008], even with simplifications such as the access-to-effect model-to-model vulnerabilities. Attack graphs are complex for humans to understand (Please refer to Appendix-E, the Figure showing the complexity of a classical AG) for two primary reasons. (1) The majority of attack graphs do not completely characterise the topology of the network, i.e. firewalls, network hierarchy, logical grouping of hosts such as subnet, LAN (Local Area Network), and VLAN (Virtual LAN) are not represented in the graph itself. Therefore, it is difficult to relate attack graphs to the network itself. (2) The overload of information is above a human's capacity, even for small networks, due to the number of arcs and the absence of logical constructs for "zooming in" or "zooming out". Solutions for the problem rely on: (i) grouping, (ii) aggregation, (iii) clustering, (iv) tree maps (as shown in Figure 1-6), and (v) prioritization, applied after the attack graph is built. [Mehta, Vaibhav, et al., 2006] suggests the grouping of nodes with similar configurations into one single node in the graph. Their reason is that such hosts may have similar vulnerabilities. They leave as a challenge to "determine how to group these subsets in an efficient manner so that there is little overlap and redundancy in the paths". Other researchers (e.g. [O'Hare, Scott et al., 2008] [Ou, Xinming et al., 2006] (in Net SPA tool) applies grouping methods to reduce reachability computation. Noel and Jajodia applies aggregation rules, visual clustering, and adjacency matrix clustering [Shuzhen et al., 2012] to overcome the complexity of attack graphs generated by the CAULDRON tool. Utilising these techniques and methods reduces the visual complicity of attack graphs, when they are utilised with a

modest number of exposures. However, when the number of hosts and vulnerabilities increases, there is a significant increase in the complexity and overload of information appeared in some instances, some of these methods were adapted to implement cost metric calculations; this will be discussed in further detail later in this thesis in Chapter-4.

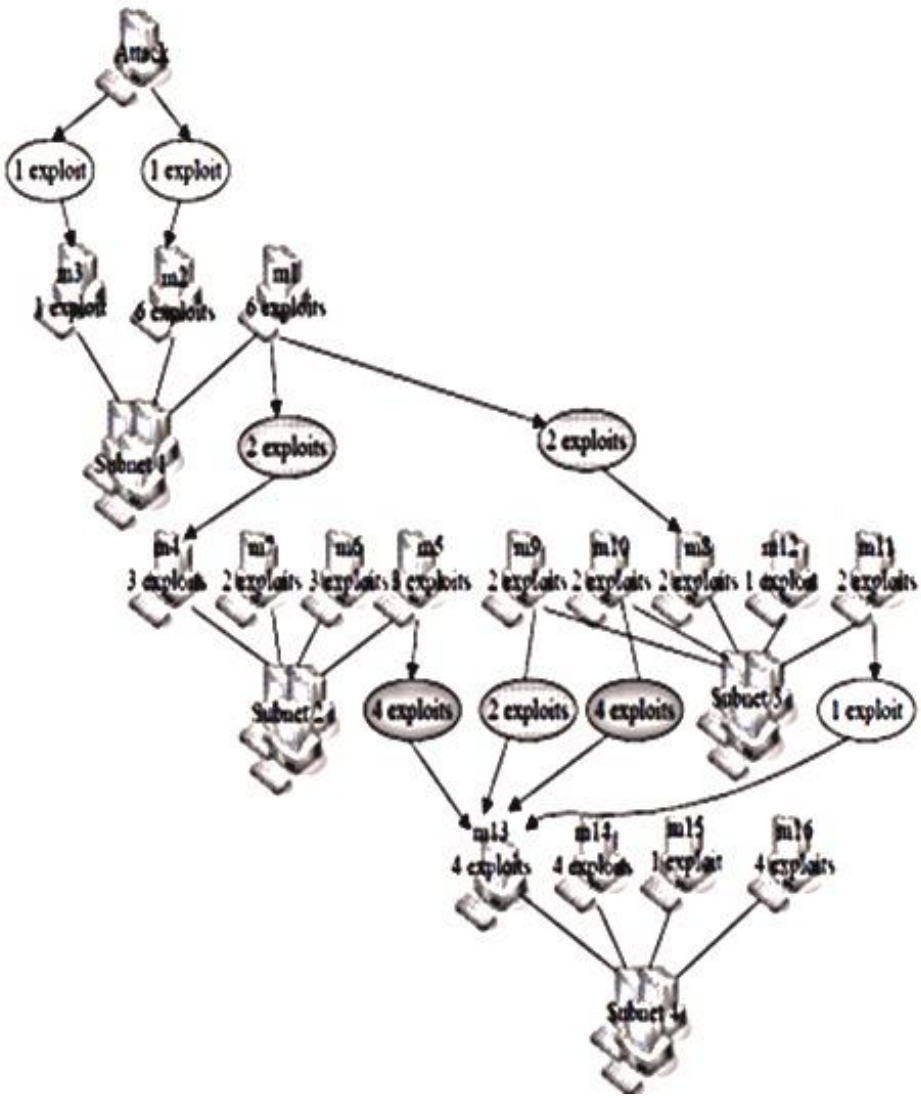


Figure 1-6. Aggregation applied to a prototype network [Franqueira, Virginia et al., 2009]

1.5 CONTRIBUTION OF THIS THESIS

The Centre of Strategies and International Studies (CSIS) July 2013 estimated criminal activity cost on cyberspace to be in the range of $300 billion to $1 trillion and estimated the cost of breaching privacy in the range of $1 billion to $16 billion. Many victims in cyberspace do not know they have been attacked and if they discover that they were attacked, they do not report it and they do not know how to measure the cost. This lack of justification (of not having a single number of costs) makes it extremely difficult to put a monetary value on a cyber-attack, to address this in this thesis the following are developed:

- A unique severity cost of all vulnerabilities existing in a network host, using novel dynamic risk impact metric methods to develop the quantitative dynamic risk impact metric, which will help in the cost/benefit analysis of security.
- A new method to construct a cost-centric model checking approach based on attack graph for dynamic cost metric calculations
- A dynamic CostRank methodology to mathematically address the optimum result of ranking attack graphs which will cut the complexity and scale down the security cost (please refer to Chapter-6 as a benchmarking model of CostRank approach with Mehta et al. 2006 and Kijsanayothin, 2010 is demonstrated in term of efficiency and cost/benefit analysis).
- An analogous parallel algorithm is developed to make a CostRank calculation for an increasing number of hosts in the networks in an efficient and scalable manner.
- Countermeasures Attack Tree algorithms to calculate the flow of the costs and mitigations across the tree to reach the goal using a dynamic impact approach to find an optimum, effective and economic solution to implementing the countermeasures.

1.6 DEFINITIONS

Definition 1: The Network Attack Graph N_g is the representation of a network graph as follows:

$$\mathbf{N_g = (S; A_r ; N, \mathcal{N})}$$

Where **S** is a nonempty set of vertices representing states, and A_r a ∈ **S** x **S** is a non-empty set of arcs, representing communication channels to achieve a possible attack steps, N ≤ **S** is a nonempty set of nodes under doubt to represent the initial position of an attacker. The $\alpha_r \in A_r \rightarrow \mathcal{N}$ is a representation of the function, which allocates a cost metric to arcs (i.e. a cost metric that an attacker comes upon to compromise an arc). S represents the number of space states.

Definition 2: If the states (S) are given to achieve and an attack (Ac), the associations required for the attack ($\alpha_n \subseteq S \bullet A_c$) and point toward associations $\alpha_p \subseteq S \bullet A_c$, the attack graph $N_g = (S \bigcup A_c, \alpha_n \bigcup \alpha_p)$, then called directed attack graph where $(S \bigcup A_c)$ represent sets of states/vertexes and $(\alpha_n \bigcup \alpha_p)$ represents sets of arcs.

Definition 3: If a possible multi-step attacks (A_c1, A_c2) are given and $a_c \in \alpha_n \bigcup \alpha_p$ where $s \in S$, then this called as a Sequence attack configuration within a directed attack graph if $A_c1 \rightarrow A_C2$

The implication of this configuration is: A_c1 happened and completed, then Ac2 started.

Definition 4: If a possible multi-step attacks (A_c1, A_c2) are given and $a_c \in \alpha_n \bigcup \alpha_p$ where $s \in S$, then this called as a Parallel attack configuration within directed attack graph if $A_c1 \Leftrightarrow A_C2$

The implication of this configuration is: A_c1 and Ac2 occur simultaneously.

Definition 5: If a possible multi-step attacks (A_c1, A_c2) are given and $a_c \in \alpha_n \bigcup \alpha_p$ where $s \in S$, then this called as Alternate attack configuration within directed attack graph if $A_c1 \Uparrow A_C2$

The implication of this configuration is: A_c1 and Ac2, but not both, occurs occurs at the same time.

Definition 6: Let **Ng = (S; Ar ; N, $\mathcal{N}$)** is a network attack graph. The set of all possible multi-step attacks A_c over $\mathbf{N_g}$ is defined as follows:

For every (A_c1, A_c2,..A_cN) and $a_c \in \alpha_n \bigcup \alpha_p$ where $s \in S$, there are ¶ (A_c1, A_c2,..A_cN) in A_c, for ¶ $\in$ { ($\rightarrow, \Leftrightarrow, \Uparrow$)}.

where $\rightarrow([A_C1 \rightarrow [A_C2, A_C3] = \rightarrow([A_C1, A_c2] \rightarrow [A_C3]) = (\rightarrow([A_C1, A_C2, A_C3])$

$\Leftrightarrow([A_C1 \Leftrightarrow [A_C2, A_C3] = \Leftrightarrow([A_C1, A_c2] \Leftrightarrow [A_C3]) = (\Leftrightarrow([A_C1, A_C2, A_C3])$

$\Uparrow([A_C1 \Uparrow [A_C2, A_C3] = \Uparrow([A_C1, A_c2] \Uparrow [A_C3]) = (\Uparrow([A_C1, A_C2, A_C3])$

for (A_c1, A_c2, A_c3).

Definition 7: Let $\mathbf{N_g}$ **= (S; $\mathbf{A_r}$; N, $\mathcal{N}$)** be a network attack graph, let the cost metric of attack v= $Cost_v \in$ **$\mathcal{N}$,** we calculate $Cost_v = OLC_i$ where OLC=Operational Level Cost and i= C(s, 0) represent the cost of vulnerabilities that enables the attacker to exploit the vulnerability from the external networks or the web with no privilege escalations. i=C (s, 1) is represent the cost of vulnerabilities, which the intruder could exploit the vulnerabilities with user account privilege or gain a user account privilege. i=C (s, 2) is represent the cost of vulnerabilities, which enables an attacker to instantly advance privileged access (system administrator).

Definition 8: Let **Ng = (S; Ar; N, $\mathcal{N}$)** be a network attack graph, if the state $x \in (S \bigcup A_c)$ is given, then let $F_w(x)$ represent the groups of forward (outgoing) links from x and let $B_k(x)$ represents the groups of (incoming) links to x. Given o$d(x) = |F_w(u)|$ represents the out degree of x.

Definition 9: Let $\mathbf{N_g}$ **= (S; $\mathbf{A_r}$; N, $\mathcal{N}$)** is a network attack graph, $\Re^{(t)}(x)$ represents the CosRank of the state $x \in (S \bigcup A_c)$ and n is the number of links point to state x and $N \in S \times S$ is the total number of states in a network representation graph $\mathbf{N_g}$.

$$\Re^{(t)}(x) = ((1-d) - \frac{n}{\sum od_i} * \frac{1}{N}) + \sum_{1}^{i} (\Re_i{}^{(t+1)} * P_R)$$ for Initial attack state

$$\Re^{(t)}(x) = (d - \frac{n}{\sum od_i} * \frac{1}{N}) + \sum_{1}^{i} (\Re_i{}^{(t+1)} * P_R)$$ otherwise.

P_R = random walker matrix entries.(please refer to chapter-3 for more information).

1.7 THESIS ORGANISATION

This thesis is divided into six chapters.

In Chapter 2, the state of art in the recent research of security analysis are presented such as the development of dynamic quantitative risk impact metric methods to evaluate the security state of a computer network; attack graphs based on dynamic security metrics analysis; parallel algorithm of PageRank calculations and finally the Attack and mitigation Tree implementations are demonstrated.

Chapter 3 describes a new method to represent a unique severity cost of the total weight of all vulnerabilities for each host existing in a network host.

A novel dynamic risk impact metric methods used to develop the quantitative dynamic risk impact metric with DVSS a modified version of CVSS scores of base, temporal and environmental metrics and dedicated database to store information about the network architectures and network devices existing in the network, which might affect the reachability and connectivity.

By combining related sub-scores and modelling the problem's parameters into a mathematical framework, this will provide us the real dynamic score taking into account the importance of the assets and patches, and remedy (countermeasures) applied for each system; a new method to construct a cost-centric model checking based on attack graph for dynamic cost metric calculations. A risk matrix is used to collaborate the risk equations. Using the framework to classify the vulnerabilities to none, user and root privileges and then finding the scores to Operational Level Cost (OLC), this classification is proved that really reflects the use of the risk matrices. The calculated values used to build a cost-centric attack graph.

In chapter 4, a new methodology is developed to represent dynamic CostRank to mathematically address the optimum solution of ranking attack graphs, using a cost-centric model checking for network security. It exercises an iterative mathematical process to calculate the maximal eigenvector of a practiced matrix with a hosts' cost values derived from designated file structures. In addition, analysis and comparison of existing results of CostRank algorithm with those of Mehta et al. and Kijsanayothin approaches respectively. The same undertaking is applied to a medium size network.

Chapter 5 presents a new parallel algorithm to calculate the CostRank of a network is introduced to implement CostRank for distributed parallel computers using multiprocessors. In order to reduce the complexity of the serial CostRank algorithm as the number of hosts in the networks are increasing. The serial CostRank algorithms need to read the source CostRank values of different states and store it in a buffer, then read the parameters in the binary graph file header, which has three components: (i) entire number of states; (ii) entire number of links and (iii) the upper limit out degree in the attack graph.

A state entry record saves the essential structural data for a state such as state ID, the out-degree, and the n forward links (Link 1, Link 2, and Link v) along with the corresponding cost for each link (C1, C2, and Cv) respectively. A parallel in-memory algorithm is selected to extensively reduce access share state stored in memory and iterations, which include table partitioning (local access), synchronization of distributed table, checkpoint/restore and load balance/ and task scheduling.

In the same manner, large-scale networks are secured that, require quick and reliable computing to calculate the ranking of enormous graphs with thousands of vertices (states) and millions or arcs. In this, the focus is on a parallel CostRank computational architecture on a cluster of PCs networked via 100 MB/s Ethernet LAN. Using a cluster of 32 computers built with Pentium Core2Duo 2.54 GHz CPU, 2GB RAM, 250GB Hard Disk interconnected with 100 MB/s Ethernet LAN, running the Linux operating system, 30 PCs for partitioning calculations, and two

masters for synchronisations to assess the complexity and scalability of the algorithms.

In particular, a partitioning of input data, graph files and ranking vectors with an appropriate load balancing technique has reduced the runtime and hence scalability of large scale parallel CostRank calculations. In this dissertation, an application case study of parallel CostRank calculations is given using one-dimensional sparse matrix partitioning on a modified research page at Stanford University. It describes the link structure of the stanford.edu-domain from a September 2002 collection and contains 281903 pages with about 2.3 million links, outcomes in a major reduction in communication overhead and in runtime. To consider its efficiency, several experiments are performed using out-vector files synthesised from the real networked data.

In chapter 6, a novel effective countermeasures solution is developed against network attacks using cost-centric model checking by identifying the set of top network risks and host and application vulnerabilities and countermeasures. In the DCAT model, attack detection and mitigations are permitted not just at the leaf node, but at the transiting nodes as well. The effects of incorporating countermeasures and attacks in the DCAT are studied, using the DVSS framework and the Nessus scanner for vulnerability detections, to construct both dynamic Cost Centric Attack Trees (CCAT) and Dynamic Countermeasures Attack Trees (DCAT). The idea of mixing countermeasures into attack trees, and more generally into directed acyclic graphs, using cost centric driven from CVSS and probability is a new approach. The main difference between CCAT and a DCAT is the CCAT represents the attacker actions using cost centric values and probability, while DCAT adds a set of vulnerability mitigations to the CCAT; the new structure reduces the possible damage by an attacker to a critical asset in term of protections. The root of the DCAT represents a critical valuable asset that the system administrators need to secure and give it special consideration. The DCAT could be described as CCAT enhanced by adding a set of mitigations to each leaf node with vulnerability. To represent optimal manners of vulnerability mitigations in DCAT, the system administrator will calculate the effective cost of possible

mitigations in an attack scenario using DCAT algorithms to reduce the total impact of a specific vulnerability.

In the next chapter, the state-of-the-art methods will discuss within the areas of the research, and compare the thesis contributes to clearly demonstrate the novel contribution that is made.

Security Is A Journey, Not A Destination.

CHAPTER 2. STATE OF THE ART

Developing CostRank attack graphs based on dynamic security metrics analysis is a novel new approach to reducing the complexity of attack graphs and finding appropriate mitigations to secure Network systems from intended attackers. The focus of this chapter will be on the literatures using the following approaches and methods: (i) dynamic quantitative risk impact metrics; (ii) attack graph security metrics; (iii) parallel algorithms and PageRank calculations and finally (iv) attack and mitigation tree implementations. These approaches and methods need online tracking to meet the aims and objectives as the research in these areas cannot be considered as completed and integrated in terms of finding a solution to attack graph visual complexity and dynamic security metrics. Using and developing existing separated and different approaches and methodologies can be considered as a vital step to completing and enhancing approaches and methodologies.

The vulnerability quantitative assessment is an essential process to evaluate the security state of a system and to enable the security professionals and decision makers to implement mitigations and patch administration to protect the system from potential attacks. The quantitative score of the risk is represented by combining the consequences of a threat (impact) with the likelihood of its exploitability.

The risk should be evaluated in terms of maximum impact on an adverse event. Microsoft Baseline Security Analyser, Nessus, OpenVAS and the Skipfish scanner are used to discover known vulnerabilities. Nevertheless, all these tools use their own formulae to evaluate the severity of vulnerabilities and mostly they evaluate an individual vulnerability without taking into account the relationships between

vulnerabilities and the effect of network devices such as routers and the firewalls in the variables used by formulae to calculate the total severity.

In this chapter, the state of art in the recent research of security analysis is presented.

A SWOT (Strength, Weaknesses, Opportunities, and Threats) analysis is practiced to provide analytical evaluations of the approaches and methods.

2.1 ATTACK GRAPH SECURITY METRICS

The classifications of attack graph security metrics shown below demonstrate the state of the art approaches:

1- Attack Graph with Probabilistic metrics: represents an attack graph where each exploit of any vulnerability has a probabilistic numerical scoring value wherever the scoring value is confirmed, it means the possibility of an attacker to exploit a specific vulnerability [L. Wang et al. 2008].

2- Attack Graph with Expected Risk metrics: AG represents a chain of probable exploits and new vulnerabilities that might affect services in the future along with the estimated scoring of the vulnerability. [M.S. Ahmed et al., 2008].

3- Attack Graph with Resistance metrics: represents the structure of separate attacks, scoring, which violate the network configuration such as Firewall rules, Router access lists and polices [L. Wang et al., 2007].

4- Attack Graph with Hosts Compromise Percentage metrics: represent a chain of scoring values of the proportion of hosts on the network that the attackers can access after the exploit in terms of privileges (none, user, administrator) used [R. Lippmann et al. 2006].

5- Attack Graph with Security Risk metrics: represent a chain of vulnerabilities using scoring values driven from the measurement of many factors calculated during the course of attack such as type of exploits, accessibility and distance [F. Chen et al., 2010].

6- Attack Graph with K-step Capability Accumulation metrics: represent the structure of vitality gained by an attacker in a multi - step attack on the network in K steps [N.C. Idika, 2010].

The AG with expected risk metric is a novel approach, but out of the scope of this study, which is relevant to develop the cost-centric AG in the future work work (see Chapter 7).

The AG with resistance metric has a limited scope; the network architectures and network device configurations have been considered in calculating the dynamic cost calculations for each vulnerability and in the dynamic cost framework.

The AG with host compromise percentage is metric approach is not relevant to this study as the compromise percentage already covered in CVSS and DVSS modes.

The AG with the security risk metric approach uses a classical method for representing the chain of vulnerabilities and is not relevant in this study.

The AG with k-step accumulation metric approach uses the structure of vitality gained by attackers in multi-step of an attack. In this thesis, the interactions between vulnerabilities with privilege considerations for each state of attack are being considered using a different methodology to formulate the cost-centric attack graph using dynamic risk impact metric methods to improve the risk scoring method and reducing the visual complexity of an attack graph.

The AG with the probabilistic metrics approach is relevant to be adapted in this study as each exploit of the vulnerabilities in the AG has a probabilistic numerical scoring value representing the dynamic cost-centric of the states for each exploit. Different techniques have been used such as dynamic impact scoring, cost-centric OLC, CostRank in this study.

2.2 SECURITY INVESTMENT ANALYSIS

In this section, the analytical approach used in the business case study presented in Chapter-6 will discuss, to assist the security investment, decision makers in the process of selecting a proper security solution based on a robust cost/benefit analysis. The security investment approach is based on Daniel Bernoulli, Expected Utility Theory (EUT) and Von Neumann-Morgenstern [Mongin, 1997], these approaches calculate the consequence of the decisions under uncertainty and risk. As in this thesis, the focus on the risk evaluation of a potential attack and finding a relevant solution of vulnerability mitigations to provide security

professionals and system administrators with techniques and tools to support security decisions, the Bernoulli equations of probability theory and related work is essential to evaluate the model.

Using the [Anderson, Ross, et al., 2012.] model, the estimated cost of an attack is calculated as follows:

1. The costs represent the expectancy of an attack, such as the costs of purchases of mitigations and implementing the policy.
2. The costs, which are represent the consequence of an attack, such as the direct and indirect losses.
3. The costs, which represent the reaction to an attack, such as recompense charges to the victims and other parties.
4. Indirect costs, such as reputational impact and the loss as consequences of individual business damage of internet transactions.

The annual loss, which represent the likely loss per period (yearly in this case), can be calculated by the integral of the possible loss probability with probability values of losses with a specific security level and without implementing a protection.

Let us assume the loss measured without implementing any security $=L_{N.}$

In addition, the loss measured when implementing a specific level of security$=L_{S.}$

Then the annual lost can calculate as follows:

$$L_{N-\text{Annual}} = \int_0^\infty x.L_N(x)dx$$

$$L_{S-\text{Annual}} = \int_0^\infty x.L_S(x)dx$$

Where x is represents the possible loss.

As you can see from Figure 2-1 (b), the expected loss using a specific level of security L_S is assigning to low function loss probability and the expected loss without security L_N is assigned to high function loss probability when no security measurements as in (a),

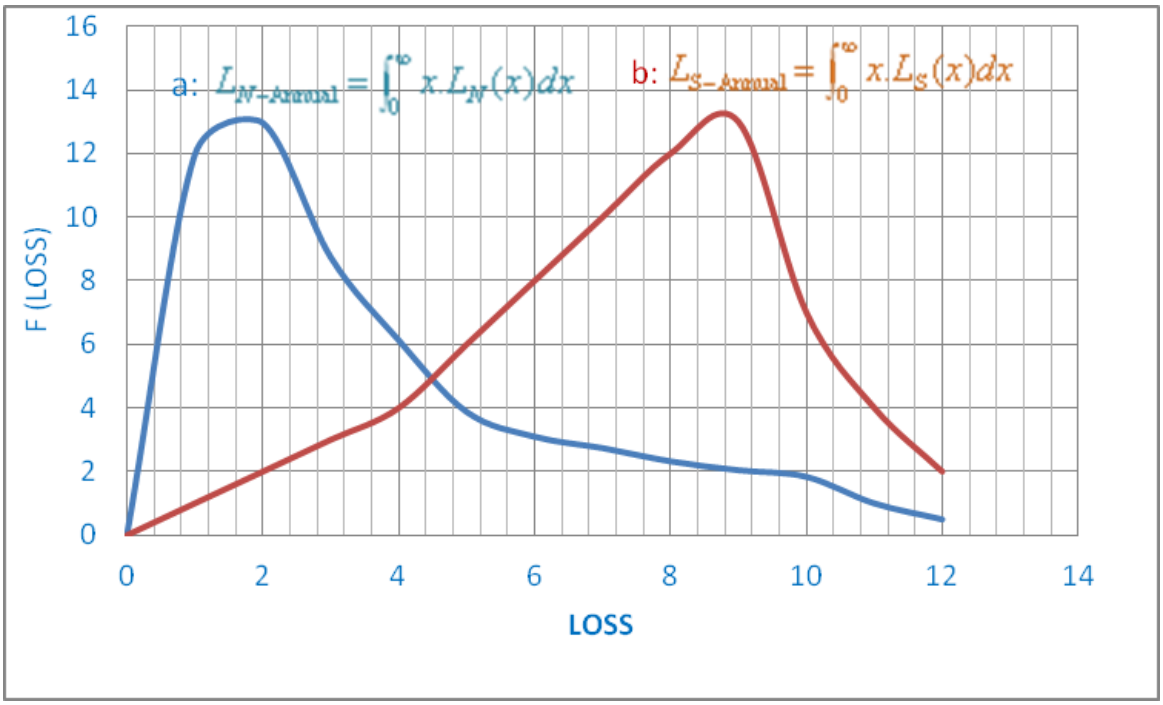


Figure 1-7. The anticipated loss of a system with (a) no Security measurement implemented; and (b) with a specific level of Security

To adjust the standard Bernoulli equations with probability theory, assume the probability of loss using the loss assumption with a specific security level=P_s, the probability of loss without implementing any security = (1-P_s) and α represents the expected impact in this case the L_S can be calculated for the hypothesis to have simple two loss outcomes (0, α) as follows:

$$L_S = P_S.\alpha + (1 - P_S).0 = P_{S.}.\alpha$$

The expected benefit (β_S) of implementing a specific level of information, security techniques can be calculated as follows Annual:

$$\beta_S = L_{N-Annual} - L_{S-Annual} = \int_0^\infty x.L_S(x)dx$$

$$\beta_S = (P_N.\alpha + (1 - P_N).0) - (P_S.\alpha) + (1 - P_S).0) = (P_N - P_S).\alpha$$

The net benefit expected ($N_{\beta s}$) can be calculated as follows:

$N_{\beta s} = \beta_s - C_S$ where C_s= the cost of implementing a level of security.

For full comprehensiveness of this approach, let us see this through a guided example:

In a company X, the security specialist suggests to implement security mitigations for existing vulnerabilities in the corporate network, with a cost of £27,645 to attack the manufacturer's Database. The Nessus scanner indicates a couple of vulnerabilities that the attackers could exploit across the Network, the report states that the probability of attack without using the security mitigations will be 40%, while the probability of the attack after using the suggested countermeasures to protect the network will be 3% only.

The estimate cost of a successful attack to exploit the manufacturer's Database Using [Anderson, Ross, et al., 2012.] model, will be £100.000.

$\alpha = 100.000, C_s = 27.645$

$P_N = 0.4, P_S = 0.01$

$L_N = P_{N.}.\alpha = 0.4x100.000 = £40.000$

$L_S = P_{S.}.\alpha + C_s = (0.03x100.000) + 27.645 = £30.645$

$\beta_S = (P_N - P_S).\alpha = (0.4 - 0.03)x100.000 = £37.000$

$N_{\beta s} = \beta_s - C_S = 37.000 - 27.645 = £9355$

Please note that in this case any N_{Bs}> 0 could be considered as a good investment.

The return on security investment ($\Re$) is defined as the ratio of the benefit of the security investment to the cost of security.

$$\Re = \frac{N_S}{C_S} = \frac{\beta_S - C_S}{C_S} = \frac{L_N - L_S - C_S}{C_S}$$

In the above example, the loss calculation does not consider temporal benefits, as the investments over time are not considered.

The net present value (P_n) considers the expected net benefit of a security investment over a future interval in the corresponding present value.

$$P_n = -C_{t=0} + \sum_{t=1}^{\infty} \frac{L_{N,t} - L_{S,t} - c_r}{(1 - D_r)^t}$$

where $C_{t=0}$ represents the unrepeatable security cost at t=0.

c_r represents the recurring cost of security in interval t.

$L_{N,t}$ and $L_{S,t}$ represents the loss expectancy of interval t.

D_r represents the discount rate.

The internal rate of return (IRR) represents the discount rate (D_r) when P_n=0, a higher rate of return means the security investment is more valuable.

The above analysis will implement to calculate the expected benefit and the net expected benefit and Internal Net Return of this approach against the standard approaches of mitigations in Chapter-6.

2.3 DYNAMIC QUANTITATIVE RISK IMPACT METRIC METHODS

The attacked organisation's computer network could cause huge impact and losses for organisations such as reputation and financial loss.

So developing accurate security metrics could specify the security state and help security professionals to develop security policies and rules to protect the organisations networks.

Vulnerability quantitative assessment is an essential process to evaluate the security state of a system and to enable security professionals and decision makers to implement mitigations and patch administration to protect systems from potential attacks [Herrmann, Debra, 2007]. Microsoft Baseline Security Analyser, Nessus, OpenVAS and Skipfish scanner are used to discover known vulnerabilities, but all these tools use their own formulae to evaluate the severity of vulnerabilities. Mostly these tools evaluate an individual vulnerability without taking into account the relationships between vulnerabilities and the impact of network devices such as routers and the firewalls on variables used to calculate the total severity. When these vulnerabilities are used in, an AG with probabilities represents the static severity to detect the vulnerable paths, attackers can use this to attack the network, and this can lead to incorrect results, which can mislead the

security professional in the process of finding the exact countermeasures and mitigations.

The commercial open framework standard CVSS [Mell et al., 2007] provides useful information about the impact and communication features of different vulnerabilities, not only considering the intrinsic features of vulnerabilities, but also taking into account the environmental influence and the progress of vulnerabilities over time. CVSS represents a widely used technique for measuring the vulnerability scores on an individual (static) model. Therefore, security professionals and system administrators could be misleading, when a specific vulnerability score is low but the attackers can still compromise a critical asset using multi-step attack techniques, as CVSS does not take into consideration the interactions between vulnerabilities and other dynamic features of the vulnerabilities.

For example, the Google Chrome Pwnium browser attacks in 2012, as the attacker Pinkie Pie managed to execute a remote arbitrary code in the browser; using six individual low scoring vulnerabilities, in a successful multi-step attack to crack the Chrome sandbox.

1. Another example to demonstrate the attackers approach used a medium CVSS Base score of 5.0 (Please refer to Appendix-B and Chapter-3, for CVSS structure and scoring details) to cause a kernel to crash and make a Linux system completely unavailable, the vulnerability CVE-2009-1758 has the following CVSS score:

CVSS v2 Base Score: 5.0 (MEDIUM):

AV:	**N**	**AC:**	**L**	**Au:**	**N**	**C:**	**N**	**I:**	**N**	**A:**	**P**

CVSS proposes the following benefits:

1- Provide a standardised score system for all vulnerabilities, as a single vulnerability management system to indicate how fast a specific vulnerability must be, validated and mitigated.

2- Deliver a clear score that enables everyone to understand the sub scores which representing the separate features used to originate the numerical value representing the score.
3- Help to specify and produces priority to different risks, according to the numerical scoring values assigned to the vulnerabilities.
4- Consider the effect of existing vulnerabilities in different environments and the progress of the vulnerabilities over time.
5- CVSS represents an open framework, which could be used to advance an application appropriate to the organisation's needs.

Although CVSS has many benefits, it has also many weaknesses to deliver a satisfactory and beneficial quantitative risk-scoring model such as:

1- A Complete Guide of CVSS V2 by First stated, "Vulnerability scoring should not take into account any interaction with other vulnerabilities. That is, each vulnerability should be scored independently." [Mell, Peter et al., 2007]
This strategy could make sense in the process of giving individual vulnerability a score by itself; it is important to acknowledge that two or further apparently inoffensive vulnerabilities could be joined to create a critical impact.
2- The methodology of combining CVSS scores of Base, Temporal and environmental dimensions provides a useful approach to measure fixed severity scores of different vulnerabilities. However, in a multi-step attack environment, the fixed scores are a problem, as the scores combining metrics do not take into account the changes in the parameters when the attacker moves from one stage to another. A novel Dynamic Vulnerability Scoring System (DVSS) based on Intrinsic, Time-based and Ecological Metric, considers the interactions between vulnerabilities. The solution is based on utilising three files, the first file contain Vulnerabilities and related static CVSS scores, the second file contains network architecture and configurations', including network devices such as routers, firewalls and switches, the third file contains the occurrences of CVSS scoring severity levels introduced from different security organisations. These files are used with relevant probabilities to calculate the dynamic cost score for each vulnerability.

3- Some Vulnerabilities have scored zero by CVSS v2 such as arbitrary site redirect flow as the variable of the vulnerability as shown below:

CVSS v2 Base Score: 0.0

AV:	N	AC:	M	Au:	N	C:	N	I:	N	A:	N

The impact of this vulnerability is to redirect specific site like google or government or commercial website to another one, which might lead to financial, or reputation damage, but the CVSS v2 scoring is null as the confidential and the integrity and availability variables scored none.

4- Some vulnerabilities scored low or medium according to their variables, in the same time it could lead to gain the root/administrator password such as cross site request forgery vulnerability which according to their variables scored as follows:

CVSS Base Score: 4.3

AV:	N	AC:	M	Au:	N	C:	N	I:	P	A:	N

Please note if the attackers gain the root/administrator access to the to the web site, they could execute arbitrary code and gain complete access to the system and fully compromise the confidentiality, integrity and availability which make the base score much higher than 4.3.

5- 5-CVSS v2 uses three levels of scoring in the range of 0.0 to 10.0, as they use a limited number of variables many vulnerabilities scored same numerical values although the risk factors are completely different. For example (i) path disclosure vulnerability existing in some website application scored as follows:

CVSS Base Score: 5.0

AV:	N	AC:	L	Au:	N	C:	P	I:	N	A:	N

(ii) The vulnerability that leads to pass through the system and read all files reachable through the web server.

CVSS Base Score: 5.0

AV:	N	AC:	L	Au:	N	C:	P	I:	N	A:	N

6- CVSS v2 uses a limited number of access vector (AV), network, adjusted network, and local. These three levels are not enough to represent the attacker's access for examples, there are no differentiations between the local attack using network account and physical attack, in the same way there are no differences between the attacker access to the wireless network and wired network.

7- The scores in CVSS should associate the scores of individual vulnerabilities into a total assessment of the security of the complete to provide effective network security metrics considering the interaction of vulnerabilities in a specified network.

The majority of the literature in the security-engineering field does not utilise dynamic scoring by taking into consideration the interaction between vulnerabilities and addressing the multi-steps attack problem into account.

As most publications try to use the CVSS as an open framework to modify the variables to address some scoring problems to get an evaluation that is more accurate, by using sub-metrics, risk, returns to the original vulnerability sub-metrics to ensure the precision such as [Houmb, 2009].

In [Singhal, Anoop, and Xinming Ou, 2009] The researchers identified two levels of network security metrics: "component metrics" which represents static scoring of vulnerabilities such as CVSS from NVD as they don't consider interactions between vulnerabilities and "cumulative metrics" which can be considered as baseline between the static metric and the interaction between the vulnerabilities. Their solution is based on using Java; XML parser with three XML format files, the first file contains Vulnerabilities and related static CVSS scores. The network architecture, including network devices such as routers, firewalls and switches in the second file, the third file contains the network host's configurations. These files used to feed attack graph with relevant probabilities for analysis, a similar approach is being used in the cost-

centric AG methodology, but the type of the files and the manner the network architectures and configurations files interact are improved as demonstrated in Chapter-3.

In [Luo, J at al., 2014] the researchers develop software vulnerability-rating approach in which they consider the analysis of the occurrences of CVSS scoring at different periods. They modified the formulae to calculate the exploitability and the impact using the sub scores. Parameters in the analysis of these occurrences are used to develop the new scoring system. They use the average of exploitability and impact sub scoring weight. While the occurrences changes the scoring will take variable values instead of constant scoring to take variable values instead of constant scoring, The vulnerability scoring calculates, in a dynamic approach at different period of time. The scoring metric method was developed for CVSS base scores only and does not consider the temporal and environmental scoring along with testing if vulnerability severity scoring variations for future work. The approaches covered in this paper have been used in developing DVSS proposed model. [Frigault et al., 2008] provide a partial solution to the problem of associating the scores of individual vulnerabilities into a total assessment of security by combining the temporal scores of individual vulnerabilities into a global rating of security of the whole network at any given time using Dynamic Bayesian Networks based model to incorporate temporal factors. In cost-centric method a different approach being used to solve the problem, according to the model needs to create cost-centric methodology, including modified base and environmental metrics.

In this thesis, a novel approaches to developing a tool set for dynamic vulnerabilities quantitative scoring is described, including network topological analysis to measure the impact of the network devices and configurations and considering the correction factor to change the score of the intrinsic metric on the final scoring value of the vulnerabilities, as the network topology and configuration change. The formula takes different scoring values from the developed dynamic database files instead of constant empirical static values

used by CVSS, to develop a unique severity cost of all vulnerabilities using dynamic risk impact metric methods. This process aimed to improve the risk scoring method.

The dynamic impact scoring system of vulnerabilities is used to advance the cost-centric method, by assigning a unique severity cost for each host. Through dividing the scores of the vulnerabilities to three main levels of privileges (i) none; (ii) user and (iii) root, and then classifies these levels into operational levels to identify and calculate the dynamic severity cost of multi-step vulnerabilities. These approaches and methods represent an important step to achieve the aim of this thesis, to secure network systems and to assist the security investment decision makers in the process of selecting a proper security solution based on the cost/benefit analysis, as the cost-centric method will be used in building and ranking a novel cost-centric attack graph.

2.4 ATTACK GRAPHS BASED ON DYNAMIC SECURITY METRICS ANALYSIS

Attack graphs (AG) are a standard representation of how multi-step vulnerabilities can interact in order to perform an attack. Usually the AG represents the structure of system states via a group of security conditions (pre- and post-conditions), the exploits of different vulnerabilities are then represented by transitions between states. Different vulnerabilities usually stay in the system, even after been discovered and exploited because of the slow release of patches and countermeasures, or due to the expensive cost of deploying the countermeasures. Frequently the organisations compromise the availability of services to maintain the business with security risk and threats, but later they discover that they are paying a very high price in terms of cost and reputation as a consequence of an attack.

In the same style of implementing SWOT analysis of the methods, will take off with the attack graphs advantages:

1- The AG can catch hidden attack circumstances undetected by an IDS or any other security tools.
2- AG helps the security professional and the security decision makers to find the best location for the IDS and security machinery to achieve fine coverage to protect critical assets within networks.
3- Helps the decision makers and security professionals to evaluate the security policies and procedures, hardware and software setting and configurations, It can also be used to anticipate the change in the different status of the hosts within the network as a result of the attackers exploiting the different vulnerabilities in the network.
4- AG can build and identify worst security case scenarios and then accordingly the security professional can set the priority to implement the security countermeasures and mitigations.

The main weaknesses in the AG are:

1- The majority of attack graphs do not completely characterise the topology of the network, i.e. firewalls, network hierarchy, logical grouping of hosts such as subnet, LAN (Local Area Network), and VLAN (Virtual LAN) are not represented in the graph itself. Therefore, it is difficult to relate attack graphs to the network itself.
2- The overload of information is above a human's capacity, even for small networks, due to the number of arcs and the absence of logical constructs for "zooming in" or "zooming out" which makes the attack graphs output very complex for humans to understand.
3- The time scale for AG algorithms for large networks appears as a problem because the time measurement for algorithm such as the one used by MIT Laboratory [Artz, 2002] is poor even when it will used for medium network.
4- The amount of information and the time needed to analysis the pre-condition and post-condition to construct the AG is enormous, for example, it required between 10 to 60 minutes to specify the pre and post- condition for a single attack [Bilar, 2003].
5- Reachability between the nodes in a computer network is appears as AG weaknesses, especially when many Firewalls and Routers and Gateways are used. Some of the security professional wrongly assumes that the reachability information exists in the outputs of the vulnerability scanners, particularly when they are used them for each subnet, to analyse the reachability issues, information is a complex task and it is impossible to determine this information using vulnerability scanners as the output represents static information and scoring of the vulnerabilities.

In [Steven Templeton and Karl Levitt, 2000] they suggest a model to represent a chain of exploits of network vulnerabilities. They create a scenario, which links the exploits in the manner that the previous exploits provides prerequisites to the advanced exploits. Scripts presented for identifying exploits and stated that their

approach can assist in determining new exploits. The researchers were not merging the attack's case study into the attack graph model.

Their approach of using attack trees to reduce the complexity is used in an OLC method to combine the scoring of the vulnerability into a cost-centric framework.

To reduce the visual complexity of attack graph that deters the humans from fully comprehending the information presented to secure Network systems from intended attackers by providing security professionals and system administrators with techniques and tools to support security decisions and to create and maintain security policies and procedures.

In [R.W. Ritchey and P. Ammann, 2001] Ritchey and Ammann custom a single attack graph case study to examine the security of heterogeneous networks in term of published exploits. The SMV model checker is used for a single scenario showing an attack on a heterogeneous network that intrudes upon the security arrangements.

In [Paul Ammann, Duminda Wijesekera et al., 2002.] they demonstrate a scalable, embedded attack graph to evade the exponential output of a classical attack graph to reduce the visual complexity. They use a dependency attack graph and they were using the exploit conditions with a monotonicity (which means, no matter what action an attacker may perform, no changes can be made) and access privilege, network reachability to give set of preconditions and post conditions as atomic Boolean elements.

The methods used in this thesis, such as access privileges, network reachability are modified from the above approaches to make them, useable in the cost-centric AG approach.

The attack graphs presented by Ammann et al. are difficult for the human to fully comprehending, the information presented, as the AG is not clear.

Dacier in [M. Dacier,1994] suggest the privileges attack graph model as each vertex in the graph represents a privilege, while the edge represent vulnerability exploits. The Dacier model is used to build state attack graphs, which represent a new and different approach in which the attackers can exploit the vulnerability such as administrator/root privilege in the target goal. He used METF as a

probabilistic scoring metric to analyse and evaluate the security status. This model suffers from the complexity of the algorithms to represent the attack graph, some of his intuitive approach is borrowed in this thesis, but the cost-centric metric is used instead of using a METF along, to solve the problem of algorithm complexity by combining individual measures into overall attack resistance measures.

In a similar fashion to [Mehta et al. 2006] and [Kijsanayothin, 2010], a framework is implemented on a test network, using the Nessus scanner to discover known vulnerabilities, implement these results and to build and represent the cost centric attack graph using ranking algorithms. However, instead of using vulnerabilities for each host, a CostRank Markov Model has developed, thereby reducing the complexity in the attack graph and reducing the problem of visibility.

In this thesis, a scalable, dynamic CostRank algorithm for ranking the AG is presented using a new approach to represent the states by implementing a dynamic cost-centric driven from statistical probability of dynamic impact scoring for the vulnerabilities.

The ranks of the states represent the importance of the states in the DCCAT. The resulting ranking represents a metric that can be used by security professionals and system administrators to make different security decisions to improve the network security based on cost/benefits.

2.5 A PARALLEL ALGORITHM TO CALCULATE THE COSTRANK OF A NETWORK

The complexity of the serial CostRank algorithm grows polynomially, as the number of hosts and vulnerabilities in the network are increased. The serial CostRank algorithm is needed to read the source CostRank values of different states and store it in a buffer. Then read the parameters in the binary graph file header and calculate the CostRank, the goal to extensively reduce the access share state stored in memory and iterations, which include table partitioning (local access), synchronization of distributed table, checkpoint/restore and load balance and task scheduling.

Parallel computing is a well-established method, and researchers have used and defined different model of algorithms to satisfy their needs.

The following models of parallel algorithms have been studied in terms of strengths and drawbacks:

1- The parallel random access memory (PRAM) model is look upon as fundamental parallel algorithm models and proposed the opportunity of solving the problem of the physical limitations of the computers using many processors connected to a shared memory to run the tasks in parallel. The focal weaknesses of PRAM model are the impractical expectations of the communication overhead and the synchronisations of the instructions cannot use the model for real time parallel computing and the complexity of the algorithm.

2- Bulk-synchronous parallel model (BSP) projected in 1990 by [L. G. Valiant, 1990]. In order to solve the weaknesses in the PRAM model, the model used a fixed number of *n* processors and memory nodes connected by a computer network, this model used the super step idea of encompassing the synchronisation and communication process, also used public variables or transient message in processor communication.

   The BSP model is supplementary representative comparing with PRAM as it is considering most of the parallel commuting overhead except the professor's management overhead.

   In the parallel PageRank, numerous methodologies have been proposed. In [S. Kamvar, T. Haveliwala, 2004], the researchers suggested an adaptive methodology by organised groups of web pages founded on the speed of convergences of PageRank ranking values, then accordingly the resources used to perform the distributed algorithm allocated. In other approaches as in [K. Avrachenkov at al., 2007] the communications are completed using multiple synchronisation servers to communicate the variable values between different processors involved in PageRank computations. In [Y. Zhu at al., 2005] the authors are suggested chunk of structure to implement Markov chain to compute the distributed PageRank.

In this thesis, most of BSP advantages are adopted, considering the process management and the scalability. A partition-centred CostRank algorithm based on BSP has suggested that can efficiently run on a parallel background. A visible logical dialogue has provided in terms of I/O and synchronization rate, and memory utilisation, speedup gain by using parallel CostRank over the Serial CostRank algorithm, load balancing, processor loads, and efficiency of parallel CostRank algorithms.

The CostRank algorithm is developed, based on the PageRank Markoven chain approach, to implement the cost-centric ranking for distributed parallel computers using multiprocessors. In order to reduce the complexity of the serial CostRank algorithm, which is grows polynomially (as it includes a matrix multiplication). Satisfactory to make CostRank calculations more effective in terms of accuracy, cost/benefit trade-off, because in the arena of security management, untreated threats can be hazardous and if the network administrators and security professionals are unaware of an attack, the impact could be very detrimental to the businesses concerned.

2.6 DYNAMIC COUNTERMEASURES ATTACK TREE ALGORITHMS

The AG is a natural application to build a security scenario graph, especially when ranking of the vertex is required and a huge number of vulnerabilities need to be chained and the considerations of multi-step attack, the relations between vulnerabilities need to be addressed. For the aforementioned reasons AG is used in developing the methodology of a dynamic CostRank attack graph. In the process of selecting, a suitable graph structure to optimise the countermeasures process the AT is selected for the following reasons:

1- Bruce Shneier's [Opel, Alexander, 2005] The AG is a natural application to build a security scenario graph as the exploits structured similar AT, Conversely the chance of cyclic dependency, the state's merged could happens inside AG.

2- In AT the leaves node could represent the vulnerabilities or a countermeasure for mitigations, in other hand the attack represented as set of actions to compromise the vulnerabilities.

3- To mitigate the risk in AT, an attacker could exploit a vulnerability in the method of OR attack, the security professional/system administrator needs to find a countermeasure for every conceivable exploit, in the method of AND attack, it will be sufficient to present a mitigation for only one conceivable exploit to protect the target asset.

The fault tree model was advanced in 1960 [Vesely et al., 1981] to assess systems, safety and maintenance, security engineering, Schneier introduced AT as tools for assessing the systems security, a clear constraint of AT that they are not integrated the relations between the countermeasures and the exploits that the attackers try to use the vulnerabilities across the system.

Defence trees (DTs) [Bistarelli, Stefano et al., 2006] have been developed to investigate the effect of defence mechanisms using measures such as attack cost, security investment cost's weight, return on attack (ROA), and return on investment (ROI) [Roy, Arpan et al., 2010]. However, placing defence mechanisms only at the leaf nodes cannot represent a complete solution; the corresponding ROI/ROA analysis does not incorporate the probabilities of attack and the attack cost used in these models did not reflect the real numerical scoring metrics for both the vulnerabilities and the mitigation.

In this research, all available techniques are used to develop a novel effective countermeasure solution against network system attacks was established using a cost-centric attack graph by developing DCAT, in the DCAT model, attack detection and mitigation are legitimate not just at the leaf node, but at the transiting nodes.

The results of incorporating countermeasures and attacks in the DCAT, using the DVSS framework and the Nessus scanner for vulnerability detections to construct both dynamic attack trees and DCAT, show clearly the effectiveness and the significance of the CAT algorithm. This is both are in terms of reducing the total impact of a specific vulnerability and giving the

security professional and system administrators the capability to select more effective mitigations.

In the following chapter, a novel approach to developing a tool for dynamic vulnerability quantitative scoring is awarded, including network topological analysis to assess the impact of net devices and forms.

The dynamic impact scoring of vulnerabilities would give the decisions maker clear idea of the requirements to secure the organisation's networks to avert financial and reputational impacts.

CHAPTER 3.DYNAMIC COST-CENTRIC RISK IMPACT METRICS

3.1 INTRODUCTION

The Oxford Dictionary [Oxford Dictionaries, 2011] defines a metric as “a system or standard of measurement.” From this meaning, the measurement is utilised to clarify the term metric. Quantitative security metrics could present a clear image of security structure, security problems, and resolutions, security policies, and procedures. Furthermore, helps the security professionals to understand the capability of a system,and to carry on a task whilst being under attack.

Risk assessment can be carried out to assess system reliability and to benchmark different security solutions. A measurable risk assessment is the primary tool used to specify if the additional budget should be allocated to afford more security features, as the mathematical numbers of the security protections and weaknesses, would give the decision maker a clear idea of the requirements to ensure the organisation's network security to avert financial and reputational impacts.

The objective of risk assessment is to measure the likelihood of exploitability and its consequences (Impacts), by measuring the probability that manifest threats in terms of access required, attack complexity and weighting the impact of the occurrence with the damage potential that may occur.

According to [Herrmann, Debra, 2007] there are three major phases of security metrics:

A. Compliance metrics: measures compliance with current security and privacy regulations and standards, such as Health Insurance Portability and

Accountability Act (HIPAA), Sarbanes-Oxley, Gramm-Leach-Bliley Act (GLBA), etc.

B. Resilience metrics: measures the resilience of controls relating to operational security and IT security, physical security, personnel security, both before and after a product, systems or network are deployed, such as the metrics are used in MITRE Establishment.

C. Return on investment (ROI) metrics: measures the ROI in physical, personnel, IT and operational security controls to guide capital investment.

National Vulnerability Database (NVD) and The Common Vulnerabilities and Exploits Database (CVE) provide specific CVSS scores for publicly known vulnerabilities, as the CVSS is becoming an industry standard and widely used in private and public sectors, is relevant to consider it as new forms of security metrics.

Different systems of security metrics exist and these have been developed by different organisations. They hold their own methodologies to use different metrics such as ROI based on capital investments; CVSS use the different parameters related to vulnerabilities, they employ different manners to measure and estimate the risk. For example, the CERT/ Coordination Centre (CC) [Jajodia et al., 2005] [Mell et al, 2007] is introduces, a metrics system which produces a numerical outputs ranging from 0 to 180. According to the numerical output values, a decision will be made to consider whether the organisation's infrastructure is at risk or not, this inconsistency in a security metrics can lead to security weaknesses issues and inaccuracies

The SANS vulnerability analysis outputs a range of numerical values to show whether there are any weaknesses in the Server or Client configurations and whether they need major/minor adjustment.

Microsoft's proprietary scoring system (MPSS) was designed to indicate a severity of compromising the different vulnerabilities during the exploit, and give some indication of the impact of the vulnerabilities.

The Common Vulnerability Scoring System (CVSS) represents open references for different vulnerabilities and their corresponding impacts [Mell et al, 2007][Papadaki, Maria, and Steven Furnell, 2010] [Purboyo et al., 2012].

The CVSS model is designed to offer the end-user with a complete combined score system symbolising an individual risk and severity of a vulnerability. It is resultant from applying specific formulae to metrics in three different groupings that could be evaluated quantitatively or qualitatively. Base Metrics comprise merits describing a specified vulnerability, which does not change in dissimilar circumstances and over time. Temporal Metrics comprise merits originalities of an exposure that has altered and changed in dissimilar circumstances and over time. Environmental Metrics comprise those individualities of a vulnerability, which are linked to an operation on a particular user's environment.
CVSS used to assess the vulnerability scores on individual (static) style as a result, the security professionals and system administrators could be misled when a specific vulnerability score is low, but the assailants can be compromised a critical asset using multi-step attack techniques. As CVSS is not taken into the considerations the interaction between vulnerabilities and other dynamic featured of the exposures. The computed values of the dynamic method used in a novel approach to building the cost-centric attack graph.

3.2 RESEARCH METHODS

Unidentified enemies with unknown skills, knowledge, resources, authority, motives, and objective attacking make risk measurement and management really challenging. This chapter contains work related to dynamic quantitative representation and investigation of computer and information security. The ability to accurately describe a security using quantitative methods could provide better control and evaluation of protection in operational settings [Herrmann, 2007]. You cannot master what you cannot assess. To recognise how well security requirements are met, one manner to approach this problem is to attempt

quantification. The goal of this work is to advance a new methodology to measure a dynamic severity cost impact for each host by developing the Common Vulnerability Scoring System (CVSS) based on base, temporal and environmental metrics to create a Dynamic Vulnerability Scoring System (DVSS) based on Intrinsic, Time-based and Ecological Metric.

The interactions between vulnerabilities are considered and dynamic impact metric is produced, which can be viewed as a baseline between the static metric and the interaction between the exposures. The solution is based on utilising three files; (i) the first file contain Vulnerabilities and related static CVSS scores. (ii) The network architecture and configurations, including network devices such as routers, firewalls, and switches in the second file, (iii) the third file contains occurrences of CVSS scoring severity levels introduced from different security organisations. These files are used with relevant probabilities to estimate the dynamic cost score for each vulnerability. This is driven from a modified network.xsl, which contains the network infrastructure and configurations to calculate a dynamic new score for each vulnerability and save it as Dynamic.xsl file. By combining related sub-scores from dynamic file, a unique severity cost is produced by modelling the problem's parameters using mathematical framework, a mathematical model is an appropriate approach as the different parameters used to calculate the final scores determined from network architecture, device setting, and the impact of vulnerabilities interactions. The individual static nature of CVSS affects the scoring value, thus a model has adapted to produce a DVSS metric that is more precise and efficient. In addition, it also addresses the problems of some vulnerability, which could have the same CVSS score even though the impact and accessibility is different. This is accomplished by introducing a correction factor, as different security organisations are susceptible to the same vulnerability; different levels of severity; the intrinsic correction factor is employed to measure the intrinsic scoring. This is reflected in the corrections.xsl file that is used in a mathematical formula to calculate the correction factor, which might have negative values to adjust the intrinsic scoring.

A new method has developed to represent a unique severity dynamic cost of the total weight of all vulnerabilities for each host to represent the cost-centric severity for each state.

A vulnerability rating approach is developed which considers the analysis of the occurrences of DVSS scoring at different periods (temporal); the formulae are modified to calculate the exploitability and the impact using the sub-score, parameters in the analysis of these occurrences to develop the new scoring system using the average of exploitability and impact sub-scoring weight.

As an example, the equation to calculate the Intrinsic metric (InS) is:

$$\overset{D}{InS} = (round^{1}((0.6 * \overset{D}{I} + 0.4 * \overset{D}{E}_{x} - 1.5) * Function(\overset{D}{I}) + \Omega)$$

where round^{1}=round to one decimal, $\overset{D}{I}$ =Dynamic Impact, $\overset{D}{E}$ =Dynamic Exploitability.

Please note $\vec{AV}, \vec{AC}, \vec{Au}$ and $\vec{CI}, \vec{InI}, \vec{AI}$ values are driven from the files specified above, which represent the dynamic values of vulnerabilities interactions and the network topologies and devices and the Calculated occurrence values.

While the occurrences changed, the scoring will take variable values instead of constant scoring. The vulnerability scoring calculated in the dynamic approach at different period.

This approach was used to solve the problem with CVSS v2 of using a limited number of variables as much vulnerability scored the same numerical values, although the risk factors are completely different that will leads to miscalculate the risk factors and influence the security policies and mitigation plans.

The DVSS framework is developed using modified equations to simplify the calculations of the vulnerability impact, dynamic scores and to benchmark against other models. The framework that could be applied by organisations to manipulate their own schemes and execute their own audits.

A novel approach to exemplifying the cost-centric to each host is projected. by dividing the scores of the vulnerabilities into three primary degrees of privileges: (i) none; (ii) user and (iii) root, and classified these levels into operational levels to

key out and compute the dynamic severity cost of multi-step vulnerabilities to produce a unique severity cost for each vulnerability states, to be used to build the cost-centric attack graph. As this approach will reduce the visual complexity of the classical AG and assessing the security professionals to detect and mitigate the vulnerabilities and provide effective protections to the systems.

This approach is used to solve many problems that exist in CVSS v2 such as: (i) the problem of the scores in CVSS, which should associate the scores of individual vulnerabilities into a total assessment of security to provide effective network security metrics that consider the interaction of vulnerabilities in a specified network. (ii) The problem existing in some vulnerability, as some of them scored low or medium according to their variables, but at the same time, it could lead to an attacker gaining the root/administrator password. (iii) The problem of some Vulnerability scored zero by CVSS v2 such as arbitrary site redirect flow and it might cause financial or reputational damage.

Finally, the framework is implemented on a prototype network; a test plan is introduced, by examining the dynamic scoring of DVSS using a practical network example.

The implementations indicate CVSS combining scores problems and testing dynamic cost framework using the Nessus Scanner as a tool to detect known vulnerabilities on a prototype network.

A check list is prepared to make sure that the framework solve the problems existing in the static model, using DVSS cost assigned to each host and the operational levels to identify and calculate the severity cost of multi-step vulnerabilities to implement a cost centric attack graph.

The initial result shows the substantial reduction of the visual complexity of the dependency cost-centric attack graph.

3.3 VULNERABILITY DATABASES

In this thesis, many examples are presented using vulnerability databases to represent and benchmarking the vulnerabilities using DVSS, CVSS scores and it is significant to make a survey of different vulnerability databases.

The Common Vulnerabilities and Exploits Database (CVE) is a cooperative effort along with the theatrical role of the MITRE Corporation and research facilities, institutions, and projects [Herrmann, Debra S, 2007]. CVE list is an initiative to standardise vulnerability references and gives vulnerabilities a name in the form CVE-XXXX-YYYY [Purboyo et al., 2012], where XXXX represents the day of the month/year in which the vulnerability was recounted/reported. This central database allows each of the vulnerabilities to have one unique identifier, a CVE id, such as “CVE- 2002-0649.” The use of unique identifiers reduces the complexity of the international security threat identifications effort, in that are fewer duplicate vulnerabilities circulating, enabling a cleaner, and simpler network report to be generated. CVE serves as more of a dictionary of vulnerabilities than a database.

The National Vulnerability Database (NVD) is one of the National Institute of Standards and Technology (NIST)’s important security assets for learning the severity of computer security risks. NVD is the core of many other security databases and utilises the CVSS scoring system, allowing the fullest use of available public computer security risk analysis and quantification methods via CVSS scores [Mell et al, 2007]. NVD is also linked with CVE, enabling comparison and the expansion of NVD with CVE entries. The CVSS scores from NVD, and identified vulnerability signatures in NVD entries, allow for this automated approach. NVD is used as the primary resource for finding vulnerabilities and determining their comparative severity and impact. Using NVD’s information about the vulnerabilities, vulnerability signatures can be derived, enabling the matching of network conditions to the extracted signatures, then marching to CVE IDs, and getting the CVSS base score from the NVD entries. Scores can be acquired for each of the vulnerabilities identified from the matching process. NVD provides a reputable, widely used, constantly updated, and openly available resource [Purboyo et al., 2012].

3.4 QUANTITATIVE RISK EVALUATION

The Risk is limited to be a mapping of the probability (likelihood) and the severity (impact) of the probable breaches on a scheme. The risk in IT systems could be exposed from the Internet, Network, Servers, and Local Host.

A prototype uses the Dynamic Vulnerability Scoring System (DVSS) metric groups: Intrinsic, Time-based, and Ecological metrics developed to measure the risk rate and safeguard a system. As discussed in Section 3.2, CVSS v2 provides an individual static scoring for vulnerabilities; this strategy could make sense in the process of measuring an individual vulnerability score. Nevertheless, it is important to acknowledge that two or more apparently inoffensive vulnerabilities could be joined to create a critical impact in a multi-step attack environment. The fixed scores are a problem, as the scores combining metrics do not take into account the changes in the parameters, when the attacker moves from one stage to another and the network architectures and configurations affects, as theses parameters will lead to ineffective scoring values of the vulnerabilities and will influence the accuracy of security decisions and polices. To solve this problem, the vulnerability scanners should combine with network resources and configurations, these resources should modify the parameters used to calculate the dynamic impact scores (DVSS).

The method as shown in Figure 3-1 can be summarised as follows:

1- Convert the Nessus scanning output to a spreadsheet XSL format (Nessus.xsl).
2- Create network architectures and configurations file Nconfig.xsl.
3- The rules sheet (rules.xsl) which represents the rules in binary to represent the influence of each network device on the reachability and other sub-scoring parameters used by CVSS v2.
4- Combine Nconfig.xsl with Rules.xsl to produce a single network specification file (Network.xsl) contains the necessary information of network architectures and configurations to be used in the DVSS calculation.

   An open source program is developed to convert Nessus scanning reports into (Nessus. xsl) is being given in that file which used to be administered against

network architectures and configurations file Nconfig.xsl and rules sheet rules.xsl to produce a single network specification file (Network.xsl).

The DVSS takes the modified $\vec{parameter}$ from network.xsl to calculate a dynamic new score for each vulnerability and save it as Dynamic.xsl.

Different organisations are dealing with security impact severity in a different manner, they are using own formulae and reporting mechanism. Such as Secunia based in Denmark is using five levels of severity, ISS X-Force, which based in USA is using three levels of severity, FrSIRT, which based in France is using four levels of severity, and CVSS, which initially was NAIC project then, now under the custodial care of FIRST-SIG is using five levels of severity. In the methodology the severity of the vulnerabilities assigned in four levels as Critical in the range of 8 -10 of base score, High if the base score in the range of 7 – 7.9, Medium if the base score in the range of 4 – 6.9 and Low if the range 0.1 -3.9 and none to support mitigation administration priorities. As you can see in the Table 3-5, the diversions of severity level, as the security organisations are given to the same vulnerability different levels of severity. To calculate the intrinsic correction factor (Ω) a corrections.xsl has been used, which contains the vulnerabilities and the corresponding scoring from above organisations, to consider the impact factors on the scoring that other organisations believes is being ignored in CVSS base calculations to make the empirical dynamic impact scoring more accurate.

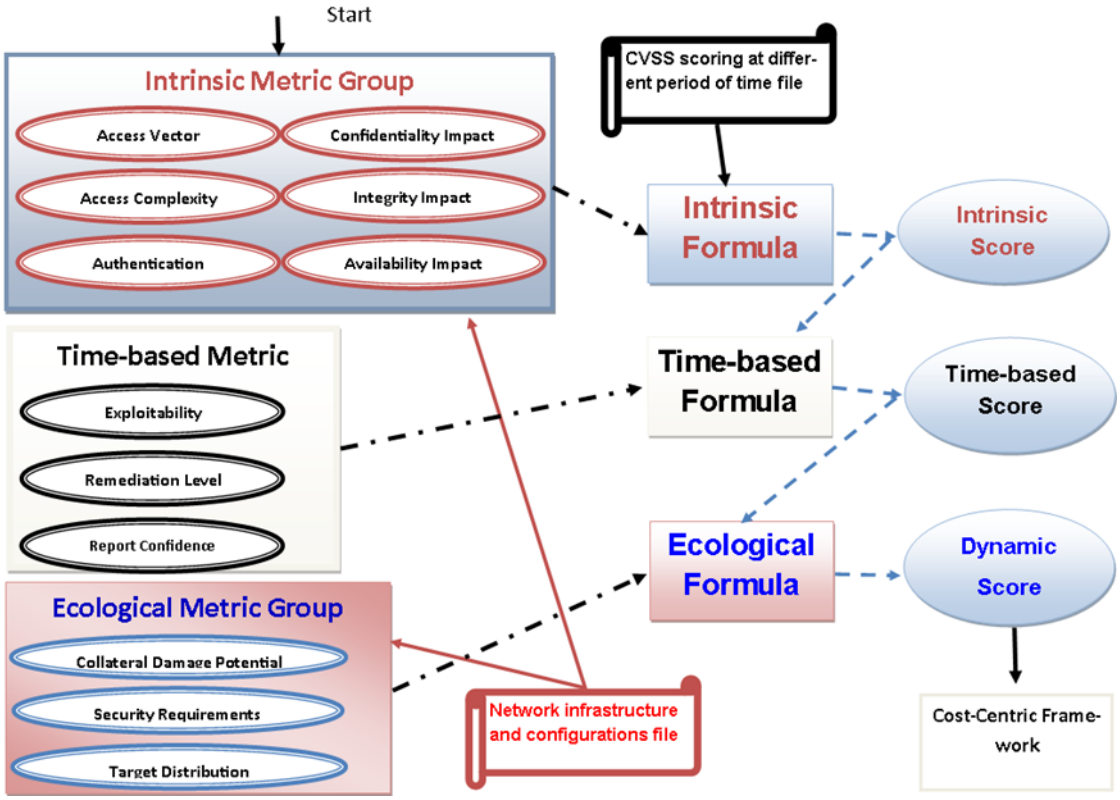


Figure 3-1. The DVSS framework combines Intrinsic, Time-based, and Ecological Metric Groups

The following formula is used to calculate the correction factor for the vulnerability x:

$$\Omega_x = CVSS - base_x - \sum_{x=1}^{n} \vec{S}_x / n$$

where S_x represents the median value for each severity level, this formula builds using the median values of the severity range used for each organisation subtracted from the static severity value provided by CVSS.

Example of calculating on the correction factor of the vulnerability CVE-2007-2242, please note the CVSS scoring range of [0-10] is utilised in the following formula, as the DVSS will convert the range to [0-100].

$$\Omega_x = CVSS_{base_x} - \sum_{x=1}^{n} \vec{S}_x / n$$

$$\Omega_x = 7.8 - ((5.45 + 4 + 6.5 + 5.45)/4)$$

$\Omega_x = 2.45$

Vulnerability	FrSIR (x/4)	Secunia(x/5)	CVSS(x/5)	Forse(x/3)	B
CVE-2007-2242	Medium	Medium	High	Medium	M
CVE-2007-3338	Critical	Medium	High	High	H
CVE-2007-1497	Medium	Medium	High	Medium	M
CVE-2007-3680	Medium	Low	High	High	M
CVE-2007-1748	Critical	High	High	High	H

Table 3-1. Security impact severity levels are offered by different organisations

The initial exploitability metrics driven CVSS base score captures access required (AV) and attack complexity (AC) and authentication instances (Au). The corresponding values can be found in Table B-1. The dynamic exploitability is calculated as follows:

The equation to calculate the Intrinsic metric (InS) is:

$$\overset{D}{InS} = (round^1((0.6 * \overset{D}{I} + 0.4 * \overset{D}{E}_x - 1.5) * Function(\overset{D}{I}) + \Omega)$$

where round^1=round to one decimal, $\overset{D}{I}$ =Dynamic Impact, $\overset{D}{E}$ =Dynamic Exploitability.

Please note $\vec{AV}, \vec{AC}, \vec{Au}$ and $\vec{CI}, \vec{InI}, \vec{AI}$ values are driven from the files specified above.

$$\overset{D}{I} = 10.41 * (1 - (1 - \vec{CI}) * (1 - \vec{InI}) * (1 - \vec{AI}))$$

$$\overset{D}{E} = 20 * \vec{AV} * \vec{AC} * \vec{Au}$$

{where $Function(\overset{D}{I}) = 0 \qquad If \qquad \overset{D}{I} = 0$

$Function(\vec{I}) = 1.176 \qquad If \qquad I \neq 0$}

The Intrinsic metric score, which represents a threat, could change over time according to the developments in exploiting methods, availability of mitigations and countermeasures for the vulnerability, exploit scripts and other information, Time-based Score (TS) is calculated as follows:

$$\mathbf{TS} = (round^{1}\ (\overset{D}{InS}_{S} * \mathbf{E} * \mathbf{R}_{L} * \mathbf{R}_{C})) * 10$$

The Adjusted dynamic Impact Score (AI) is calculated as follows:

$$\overset{D}{AI} = \text{Minimum}(10, 10.41 * (1 - (1 - \vec{CI} * \vec{CR}) * (1 - \vec{InI} * \vec{IR}) * (1 - \vec{AI} * \vec{AR}))$$

These equations were developed/adapted from the original CVSS scoring system, as CVSS is widely utilised and significantly accurate to calculate static scoring, the passing dynamic parameters, which looked at the network topologies, devices and interactions between vulnerabilities is the main criteria to calculate the dynamic scores.

The Adjusted Intrinsic score (InS_E) is recalculated using Adjusted Impact ($\overset{D}{AI}$) instead of Impact as shows below:

$$InS_{E} = round^{1}((0.6 * \overset{D}{AI} + 0.4 * \overset{D}{E}_{x} - 1.5) * Function(\overset{D}{I}))$$

The Adjusted Time-based Score (AT) is calculated using BS_E as follows:

$$\overset{D}{AT} = round^{1}\ (\overset{D}{InS}_{\mathbf{E}} * \mathbf{E} * \mathbf{R}_{L} * \mathbf{R}_{C})$$

The Ecological Metric Score (ES) could modify the metrics score of vulnerabilities, according to CDP, TD, and security requirements. The primary ground for using the quantitative metric method is to prepare a unique cost-centric value for each vulnerability; the environmental score will represent the Cost value for a vulnerability.

At this stage, a careful investigation of the formula applied is needed to calculate the ecological score, which is shown below:

$$\overset{D}{ES} = round^{1}((\overset{D}{AT} + (10 - \overset{D}{AT}) * (\vec{CDP} - 0.5) * (\vec{TD} - 1))$$

NIST [Mell et al., 2007] states that the formula used to calculate the environmental score should not produce a numerical score greater than the temporal score and the temporal formula should not produce a score higher than Base score, which represents the impact and three possibilities to exploit a vulnerability. For that reason in ES formula, they subtract the maximum CDP, TD scores from the actual CDP, TD values.

The threat of an existing vulnerability will have different impacts, according to assets essential and network settings. For example, CDP might evaluate the financial damage caused by exploiting a vulnerability in a specific asset; TD measures the percentage of assets that could be impacted if the vulnerability has been exploited. For the above grounds, the environmental score might be larger than the temporal and base scores according to CDP, TD and security requirements, the CVSS scores for base, temporal and environmental scores range from 0-10, as the cost of vulnerabilities is evaluated in DVSS from 0-100, the final score is multiplied by 10.0 to adjust the final grade.

From now on, the ES is called as $Cost_v$ as shown in the modified formula:

$$\overset{D}{Cost}_v = (round^{1}(\overset{D}{AT} + (10 - \overset{D}{AT}) * (\vec{CDP}) * (\vec{TD})) * 10.0$$

DVSS is used based on CVSS v2, to produce unique quantities metric to evaluate the static severity cost of a specific vulnerability. The methodology based on the following facts driven from a careful consideration of different CVSS sub-metrics of Base, Temporal, and Environmental.

The possible adjusted impact essential to a vulnerability (the Intrinsic metrics indicate the points of impact), could be decreased or increased based on the security needs, driven from Ecological metrics as shown below:

$$\text{Confidentiality requirement } (\text{CR})_{\text{Ecological}} \Rightarrow \text{Confidentiality impact } (\text{C})_{\text{Intrinsic}}$$

$$\text{Integrity requirement } (\text{IR})_{\text{Ecological}} \Rightarrow \text{Integrity impact } (\text{I})_{\text{Intrinsic}}$$

$$\text{Availability requirement } (\text{AR})_{\text{Ecological}} \Rightarrow \text{Availability impact } (\text{A})_{\text{Intrinsic}}$$

$$\text{Collateral damage potential } (\text{CDP})_{\text{Ecological}} \Rightarrow \text{Availability impact } (\text{A})_{\text{Intrinsic}}$$

$$(\text{TD})_{\text{Ecological}} \Rightarrow (\text{A})_{\text{Intrinsic}}, (\text{I})_{\text{Intrinsic}}, (\text{C})_{\text{Intrinsic}}$$

The primary causes behind these decisions to make the DVSS calculations more effective and accurate as the influences of Ecological sub-scores to the intrinsic sub-scores used by DVSS are considered.
According to ISO 31010 [Massacci et al., 2011] the risk combines the consequences of a threat (impact) with the likelihood of its exploitability thus:

Risk = Likelihood of an adverse event x Impact of the adverse event.

By establishing, the formulae based on the Time-based and Ecological sub metrics provided by the DVSS, which could regularly assess, update, and easily maintain the result using a cost metric approach shown in Figure 3-1.
The risk weight could be measured as a function of Impact and likelihood.

$$W_i = f(I, L)$$

where I= Impact, L= Likelihood.

A risk matrix Figure 3-2 linking the two vectors' likelihood and impact is a graphical description of different dangers in a comparative manner. The matrix uses the standard four levels of weights to classify the ranks of different cases of risks based on empirical grounds.

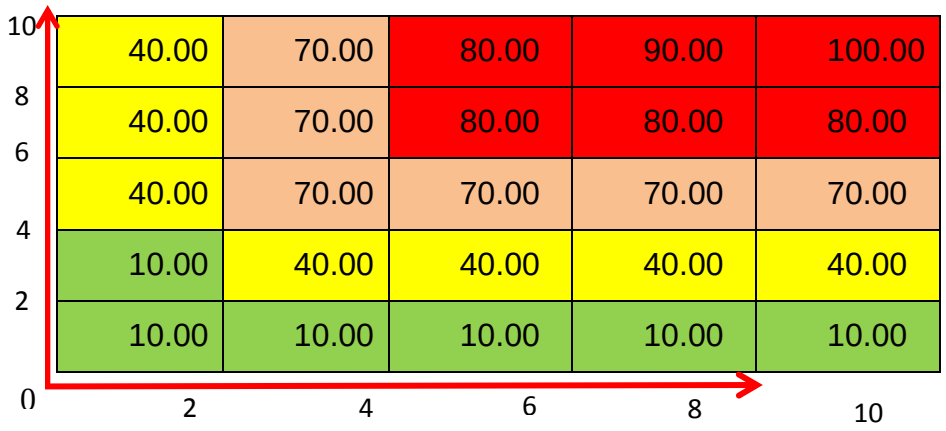


Figure 3-2. A risk matrix, provides a description of the hazard ranking

1- Level 1 (80-100) (Critical cost): Exploiting a vulnerability that allows an attacker directly to gain privileged rights to access (Administrator, root, power user) the system. For example, using a Buffer overflow attack, from a local or remote system to gain root access to a specific system.
2- Level 2 (70): (High cost): Exploiting a vulnerability that allows local or remote users to increase their privileges on a system or access confidential information, such as company financial records or user passwords, are usually considered moderate risks.
3- Level 3 (40) (Medium cost): Exploiting a vulnerability like a Denial-of-Service attack: Generally, these do not compromise the system beyond a Denial-of-Service. This type of attack often prevents running a particular service.
4- Level 4 (10) (Low cost): Exploiting a vulnerability that provides information, (Information about the target is gathered) to an attacker who might use them to carry out additional compromise efforts.

This model is widely used in most risk assessments; some other models just use low, medium and high. This model is suitable for the cost-centric approach as it represents the risk in terms of Critical risk, high, medium, low and none.

The DVSS framework is built, using modified equations to simplify the calculations of the vulnerability scores and to benchmark against other models. The DVSS scores range from [0, 100] with the higher score meaning worse security.
The following examples are to test the DVSS framework, by evaluating the sub-scores attribute with the Ecological (Cost) score, in example 2-1, the attributes of maximum sub-scores values should result in a maximum Ecological score and vice versa in example 2-2.

Example: 2-1 (**Critical cost-Maximum**)

If a given vulnerability has the following DVSS sub-scores (maximum) attributes, then the Ecological score represents a critical cost:

Intrinsic:

AV:	**N**	**AC:**	**L**	**Au:**	**N**	**C:**	**C**	**I:**	**C**	**A:**	**C**

Time-based:

E:	**H**	**RL:**	**U**	**RC:**	**C**

Ecological($Cost_v$)

TD:	**ND**	**CDP:**	**H**	**CR:**	**H**	**IR:**	**H**	**AR:**	**H**

The equation to calculate the Ecological ($Cost_v$) is:

*As a replacement for Ecological scores, will be used as Cost v, as this account represents the final dynamic impact cost for vulnerabilities.

1- The Impact (I) and Exploitability (E) are calculated as follows:

$$\overset{D}{I} = 10.41*(1-(1-\vec{CI})*(1-\vec{InI})*(1-\vec{AI}))$$

I=10.41 * (1-(1-0.66)*(1-0.66)*(1-0.66))

I = 10.

$$\overset{D}{E}_x = 20*\vec{AV}*\vec{AC}*\vec{Au}$$

$$\overset{D}{E}_x = 20*1.0*0.71*0.704$$

$$\overset{D}{E_x} = 10.$$

2- The Intrinsic score calculate as follows:

$$\overset{D}{InS} = (round^1((0.6 * \overset{D}{I} + 0.4 * \overset{D}{E_x} - 1.5) * Function(\overset{D}{I}) + \Omega))$$

$$\overset{D}{InS} = (round^1((0.6 * 10.0 + 0.4 * 10 - 1.5) * 1.176) + 0))$$

$$\overset{D}{InS} = 10.0$$

3- The Time-based Metric Score (TS) is calculated as follows:

$$\overset{D}{\mathbf{TS}} = (round^1\ (\overset{D}{InS_S} * \mathbf{E} * \mathbf{R}_L * \mathbf{R}_C))$$

$$\overset{D}{\mathbf{TS}} = (round^1\ (10.0 * 1.0 * 1.0 * 1.0))$$

$$\overset{D}{\mathbf{TS}} = 10.0$$

4- The $Cost_v$ is calculated as follows:

The Adjusted Impact Score (AI) is calculated as follows:

$$\overset{D}{AI} = \text{Minimum}(10, 10.41 * (1 - (1 - \vec{CI} * \vec{CR}) * (1 - \vec{InI} * \vec{I}\,\vec{R}) * (1 - \vec{AI} * \vec{AR}))$$

$$\overset{D}{AI} = \text{Minimum}(10, 10.41 * ((1 - (1 - 0.66 * 1.51) * (1 - 0.66 * 1.51) * (1 - 0.66 * 1.51))$$

$$\overset{D}{AI} = 10.0$$

The Intrinsic score is recalculated to get the Adjusted Intrinsic score (InS_E) using Adjusted Impact (AI) instead of Impact as shown below:

$$\overset{D}{InS_E} = round^1((0.6 * \overset{D}{AI} + 0.4 * \overset{D}{E_x} - 1.5) * Function(\overset{D}{I}))$$

$$\overset{D}{InS_E} = round^1((0.6 * 10.0 + 0.4 * 1.0 - 1.5) * 1.176)$$

$$\overset{D}{BS_E} = 10.0$$

The Adjusted Temporal Score (AT) is calculated using BS_E as follows:

$$\overset{D}{AT} = round^{1}\ (\overset{D}{InS}_{\mathbf{E}} * \mathbf{E} * \mathbf{R}_{L} * \mathbf{R}_{C})$$

$$\overset{D}{AT} = round^{1}\ (10.0 * 1.0 * 1.0 * 1,0)$$

$$\overset{D}{AT} = 10.0$$

The $Cost_v$ Score is calculated as follows:

$$\overset{D}{Cost}_{v} = round^{1}((\overset{D}{AT} + (10 - \overset{D}{AT}) * (\vec{CDP} - 0.5) * (\vec{TD} - 1)) * 10$$

$$\overset{D}{Cost}_{v} = round^{1}((10.0 + (10 - 10) * 0.5 * 1.0)) * 10$$

$$\overset{D}{Cost}_{v} = 100, \qquad ES = 10.0$$

Example 2-2 (Low cost-Minimum) If a given vulnerability has the following DVSS sub-scores (minimum) attributes, then the Ecological score represents a low cost:

Intrinsic:

AV:	L	AC:	H	Au:	M	C:	N	I:	N	A:	N

Time-based:

E:	U	RL:	TF	RC:	UC

Ecological

TD:	N	CDP:	N	CR:	L	IR:	L	AR:	L

The equation to calculate $Cost_v$ (which represents a minimum cost) is:

1- The Impact (I) and Exploitability (E) are calculated as follows:

$$\overset{D}{I} = 10.41 * (1 - (1 - \vec{CI}) * (1 - \vec{InI}) * (1 - \vec{AI}))$$

$$\overset{D}{I} = 10.41 * (1 - (1 - 0.0) * (1 - 0.0) * (1 - 0))$$

$$\overset{D}{I} = 0$$

$$\overset{D}{E}_x = 20 * \vec{AV} * \vec{AC} * \vec{Au}$$

$$\overset{D}{E}_x = 20 * 0.395 * 0.35 * 0.45$$

$$\overset{D}{E}_x = 1.2$$

2- The Intrinsic score calculate as follows:

$$\overset{D}{InS} = (round^1((0.6 * \overset{D}{I} + 0.4 * \overset{D}{E}_x - 1.5) * Function(\overset{D}{I}) + \Omega)$$

$$\overset{D}{InS} = (round^1((0.6 * 0 + 0.4 * 1.2 - 1.5) * 0 + 0))$$

$$\overset{D}{InS} = 0$$

3- The Time-based Metric Score (TS) is calculated as follows:

$$\overset{D}{\mathbf{TS}} = (round^1\ (\overset{D}{InS}_S * \mathbf{E} * \mathbf{R}_L * \mathbf{R}_C))$$

$$\overset{D}{\mathbf{TS}} = (round^1\ (0 * 0.85 * 0.9 * 0.9))$$

$$\overset{D}{\mathbf{TS}} = 0.0$$

5- The $Cost_v$ is calculated as follows:

The Adjusted Impact Score (AI) is calculated as follows:

$$\overset{D}{AI} = \text{Minimum}(10, 10.41 * (1 - (1 - \vec{CI} * \vec{CR}) * (1 - \vec{InI} * \vec{IR}) * (1 - \vec{AI} * \vec{AR}))$$

$$\overset{D}{AI} = \text{Minimum}(10, 10.41 * (1 - (1 - 0.0 * 0.5) * (1 - 0.0 * 0.5) * (1 - 0.0 * 0.5))$$

$$\overset{D}{AI} = 0.0$$

The Intrinsic score is recalculated to get the Adjusted Intrinsic score (InS_E) using Adjusted Impact (AI) instead of Impact as shown below:

$$\overset{D}{InS}_E = round^1((0.6 * \overset{D}{AI} + 0.4 * \overset{D}{E}_x - 1.5) * Function(\overset{D}{I}))$$

$$\overset{D}{InS}_E = round^1((0.6*0.0+0.4*1.2-1.5)*0.0)$$

$$\overset{D}{InS}_E = 0.0$$

The Adjusted Temporal Score (AT) is calculated using BS_E as follows:

$$\overset{D}{AT} = round^1\ (\overset{D}{InS}_{\mathbf{E}} * \mathbf{E} * \mathbf{R}_L * \mathbf{R}_C)$$

$$\overset{D}{AT} = round^1\ (0.0 * 0.85 * 0.9 * 0.9)$$

$$AT = 0.0$$

The $Cost_v$ Score is calculated as follows:

$$\overset{D}{Cost}_v = round^1((\overset{D}{AT}+(10-\overset{D}{AT})*(\vec{CDP}-0.5)*(\vec{TD}-1))*10$$

$$Cost_v = round^1((0.0+(10-0.0)*0.0*0.0)+0)*10$$

$$Cost_v = 0, \qquad ES = 0$$

3.5 CVSS v2 SCORING PROBLEMS

The DVSS solved the following CVSS v2 scoring problems, (i) the methodology of combining CVSS scores of Base, Temporal and environmental provides a useful approach to measure fixed severity scores of different vulnerabilities. But in a multi-step attack environment, the fixed scores are a problem, as the scores combining metrics do not take into account the changes in the parameters when the attacker moves from one stage to another (as evidenced by the DVSS approach that considers the reactions between vulnerabilities in the way the example demonstrated in this chapter). (ii) Some Vulnerability have scored zero by CVSS v2 such as arbitrary site redirect flow as the variable of the vulnerability as shown below:

Base score=4.3

AV:	N	AC:	M	Au:	N	C:	N	I:	P	A:	N

The impact of this vulnerability is to redirect specific site like Google or government or commercial website to another one, which might lead to financial or reputation damage, but the CVSS v2 scoring is null as the confidential and the integrity and availability variables scored none.

Exactly how the DVSS solves the problem of measuring fixed severity scores of different vulnerabilities is demonstrated here. As explained earlier in this chapter, when multi-step attacks used, the fixed scores are a problem, as the scores combining metrics do not take into account the changes in the parameters when the attacker moves from one stage to another.

The scores in CVSS should associate the scores of individual vulnerabilities into a total assessment of the security of the complete set in order to provide effective network security metrics that consider the interaction of vulnerabilities in a specified network, which introduce practical useful techniques for detecting and analysing vulnerabilities to secure network systems.

To explain this indication in detail, the network example shown in Figure 3-3 is used, which represents an attacker through external networks wanting to exploit the vulnerability in Host-3 using multi-steps attack through Host-1 and Host-2, as the parameters of the vulnerability in the host-2 establishes the Access requirement, Av=Local and the attackers based behind the firewall.

The detail of the descriptions of the vulnerabilities in Host-1, Host-2, and Host-3 is shown below:

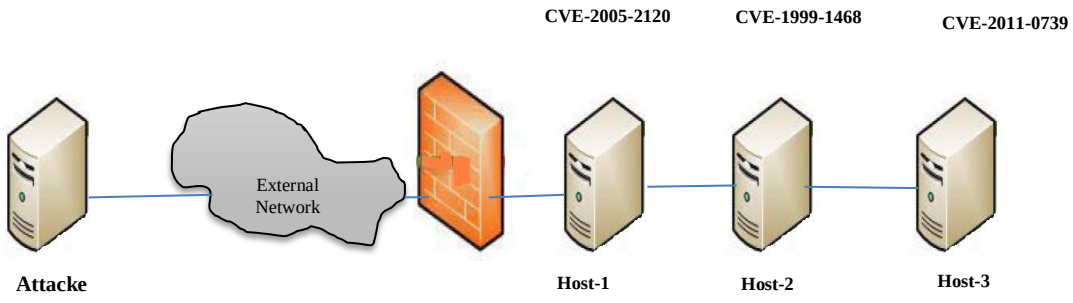


Figure 3-3. Practical Example indicates a CVSS' combining scores problems.

The vulnerabilities details shown below:

Host-1: CVE-2005-2120, this vulnerability exists in Windows XP and Windows 2000 up for services pack-4 and permits attackers to influence the plug and play services by inserting illogical script, through a huge number of backslashes into the Registry Key/System Registry. The vulnerability will lead to compromise the confidentiality partially by revealing some of the data and information, and the integrity by letting the attacker alter the data, and the availability by distressing the services.

The CVSS v2 Parameters:

Base= 6.5:

AV:	N	AC:	L	Au:	S	C:	P	I:	P	A:	P

Temporal= 5.7

E:	H	RL:	OF	RC:	C

Environmental $_v$= 74.0:

TD:	H	CDP:	MH	CR:	M	IR:	M	AR:	M

Host-2: CVE-1999-1468, this vulnerability exists in UNIX hosts, run in sending mail services and enables the local user to root to escalate the right to gain administrator/root privilege.

The CVSS v2 Parameters:

Base=6.2:

AV:	L	AC:	H	Au:	N	C:	C	I:	C	A:	C

Temporal= 5.4

E:	H	RL:	OF	RC:	C

Environmental$_v$ = 68.0:

TD:	M	CDP:	MH	CR:	H	IR:	H	AR:	H

Host-3: CVE-2011-0739, this vulnerability permits the running of an illogical script in a mail sent to a specific email address; exploiting the vulnerability will lead to compromise of the confidentiality by revealing the data and information, and the integrity by letting the attacker alter the data, and the availability by distressing the services.

The CVSS v2 Parameters:

Base= 6.8:

AV:	N	AC:	M	Au:	N	C:	P	I:	P	A:	P

Temporal= 5.9

E:	H	RL:	OF	RC:	C

Environmental $_v$= 71.0:

TD:	MH	CDP:	M	CR:	M	IR:	M	AR:	M

When the attacker manages to exploit the vulnerability in Host-1, he also manages to gain access to the LAN in the second stage of multi-step attack. The attacker will try to exploit the vulnerability in Host-2 which is located on the same subnet as the access requirement, Av=Local. If the attacker is successful, then effectively the attack has happened through the external network where the attacker is based. For that reason, the Av should replace to network instead of local and that will change the final scores of Intrinsic and the corresponding scores of Time-based and Ecological. The same consideration should be taken to calculate other parameters, for example, if in the first stage of attack the attacker managed to gain the root/administrator privileges then logically the Impact and confidentiality and availability should change accordingly. The new scores of the vulnerability in Host-2 according to DVSS are shown below:

$$\overset{D}{InS} = (round^{1}((0.6 * \overset{D}{I} + 0.4 * \overset{D}{E}_{x} - 1.5) * Function(\overset{D}{I}) + \Omega)$$

$$\overset{D}{InS} = 7.3$$

$$\overset{D}{\mathbf{TS}} = (round^{1}\ (\overset{D}{InS}_{S} * \mathbf{E} * \mathbf{R}_{L} * \mathbf{R}_{C}))$$

$$\overset{D}{\mathbf{TS}} = 6.1$$

The $Cost_v$ is calculated as follows:

The Intrinsic score is recalculated to get the Adjusted Intrinsic score (InS_E) using Adjusted Impact (AI) instead of Impact as shown below:

$$\overset{D}{InS}_{E} = round^{1}((0.6 * \overset{D}{AI} + 0.4 * \overset{D}{E}_{x} - 1.5) * Function(\overset{D}{I}))$$

The Adjusted Temporal Score (AT) is calculated using BS_E as follows:

$$\overset{D}{AT} = round^{1}\ (\overset{D}{InS}_{\mathbf{E}} * \mathbf{E} * \mathbf{R}_{L} * \mathbf{R}_{C})$$

The $Cost_v$ Score is calculated as follows:

$$\overset{D}{Cost}_{v} = round^{1}((\overset{D}{AT} + (10 - \overset{D}{AT}) * (\vec{CDP} - 0.5) * (\vec{TD} - 1)) * 10$$

$$Cost_v = 75, \qquad ES = 75$$

Intrinsic = 7.3:

AV:	N	AC:	H	Au:	N	C:	C	I:	C	A:	C

Time-based = 6.1:

E:	H	RL:	OF	RC:	C

$Cost_v$= 75:

TD:	M	CDP:	MH	CR:	H	IR:	H	AR:	H

3.6 DYNAMIC COST CALCULATION FRAMEWORK

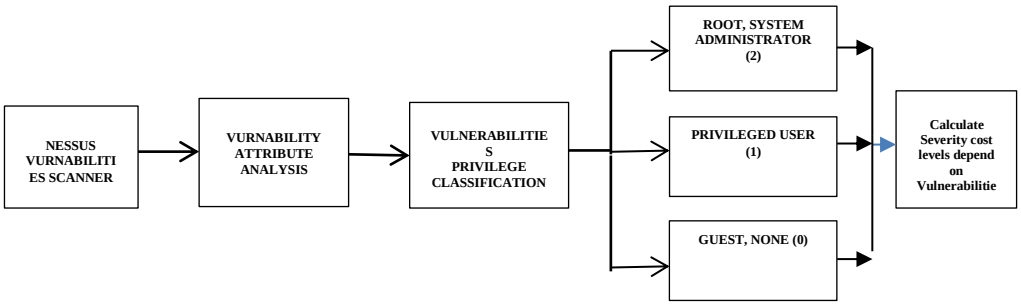


Figure 3-4. Cost-centric calculation framework using the Nessus scanner

Some vulnerability are scored low or medium according to their variables, which, at the same time, could lead to an attacker gaining the root/administrator password by using an attack such as cross site request forgery vulnerability.

Please note if the attackers gain the root/administrator access to the to the web site, they could execute arbitrary code and gain complete access to the system and fully compromise the confidentiality, integrity and availability which make the base score much higher than 4.3.

CVSS v2 used three levels of scoring in the range of 0.0 to 10.0, as they use a limited number of variables many vulnerabilities scored same numerical values although the risk factors are completely different. For example, a path disclosure vulnerability existing in any website application and the vulnerability that leads to pass through the system and read all files reachable through the web server.

The following framework designed to solve above problem and develop a dynamic impact cost-centric.

In this research, the Nessus scanner is used as a tool to discover known vulnerabilities; Nessus scanners [Deraison, Renaud, 2006] are designed to identify vulnerabilities before the attacker can exploit them as shown in Figure 3-4. Then the vulnerability attributes are analysed and classified to calculate vulnerability costs, according to privileges as follows:

1- The cost of vulnerabilities, which enables an attacker to instantly advance privileged access (system administrator. Root (2)).
2- The cost of vulnerabilities, which the intruder could exploit the vulnerabilities with user account privilege or gain a user account privilege (privileged user (1)).
3- The costs of the exploits that are enable an attacker, to exploit the vulnerability from external networks or website, with no privilege escalations. (Guest or None (0)).

The classification is done according to the vulnerability specification to User privilege vulnerabilities and root privilege vulnerabilities, which include gaining privileged access along with other impacts and none. To calculate the operational values for each group of vulnerabilities, the Attack tree structure was used to calculate the OLC (the representation of cost-centric for each state). An Attack Tree (AT) is a multiple level diagram containing one root, multiple leaves, and multiple children. As with the bottom up model, child nodes must satisfy the conditions to mark true the straight parent node. The attack is complete when the root is fulfilled. The child, through node only may satisfy the parent node. ATs with AND & OR refinement, AND means the attacker should compromise all leaf node vulnerabilities to reach the goal or to progress to the next level of attack, while OR refinement means at least one vulnerability should be compromised to reach to the goal or to progress to the next level of attack [Edge, Kenneth S., 2007]. In this model, the AG is represented by cost centric driven from CVSS as a new approach to build effective AG. Exploiting more than one vulnerability in one asset is possible in terms of AND and OR refinement associations between vulnerabilities in different time slots, to calculate the effective cost according to operational levels. Only one vulnerability from an operational level group can be exploited at a given time and based on this fact the relationships between vulnerabilities in terms of calculating the effective cost for each operational level will use only OR refinement to adjust the operational costs to be near to the

higher cost vulnerability in the group. Chapter-6 in this thesis will give the formal definitions of CCAT and CAT.

The following formulae will be used to calculate the Operational Level Cost (OLC):

$$\text{Probability}_{\text{v}} = 1 - (\prod_{k=1}^{n} (1 - P_k)$$

$$OLC_V = Max(C) - ((\sum_{k=1}^{n-1} p_k * C_k) / (\sum_{k=1}^{n} p_k /100))$$

By estimating the effective costs using AT structure, the probability and the dynamic cost of the vulnerabilities are distributed across the nodes as will be presented in details examples later in this chapter.

3.7 FRAMEWORK TEST PLAN

In this section, the foundational goal of the dynamic cost calculation framework is discussed, which is used when creating a dynamic cost-centric scoring for each status of hosts in the network and use them to build an impact cost-centric AG to reduce the visual complexity of a classical AG.

The in-depth test plan should offer a complete list of required results, functional requirements that can contribute to finding the primary strength and weaknesses of the framework.

The test plan checklist:

1- The method used in the framework to proceed analyse of the vulnerability attributes and classify and calculate vulnerability costs, according to privileges should provide effective cost according to operational levels.

2- The framework is capable to provide dynamic cost-centric for each state of the host, and classify and calculate the dynamic vulnerability costs according to privileges.

3- The proposed method should produce a security standard solution and met the system specification.

4- Inspection the cases of using a different techniques and structures used in the method could lead to inappropriate results.

5- Examining the cases when the correct setup of the test platform could lead to inappropriate outcomes and the adverse disturbs on the setup goal.

3.8 PRACTICAL APPROACH TO DEVELOPING A COST-CENTRIC ATTACK GRAPH

For the purpose of framework analysis, a prototype network setup is been used, with two computers (this approach is applicable for much more complex enterprise level network architectures as you will see in Chapter-4). PC1 (Windows XP platform with IP: 192.168.153. 160) and PC2 (Lunix Platform with IP: 192.168.153. 31) are connected with 100 MBPS Ethernet switch, the Nessus 4.x scanner installed on PC1 and program has been used to convert the Nessus scanning output to a spreadsheet XSL format (Nessus.xsl) and network architectures, and configurations file Nconfig. xsl, the rules sheet (rules.xsl) which represents the rules in binary to represent the influence of Ethernet switch configuration on the reachability and other sub-scoring parameters used by CVSS v2. An open source program was developed to convert Nessus scanning reports into (Nessus. xsl) which used to be administered against network architectures and configurations file Nconfig.xsl and rules sheet rules.xsl to produce a single network specification file (Network.xsl).

The DVSS takes the modified data from network.xsl to calculate a dynamic new score for each vulnerability and save it as Dynamic.xsl.

Instead of specifying a state by network attributes, a dynamic cost-centric model is proposed; each state in the graph is stated using the features/vulnerabilities of an individual host. The dynamic cost centric attack graph is a reference of this model by using the DVSS cost calculator and attacker privileges. The methodologies and all of the concepts mentioned are used to develop dynamic quantitative risk evaluation metrics and the existing results being used to build the dynamic cost-centric attack graphs. Nessus 4.x scanner is been used as a tool to discover known vulnerabilities on PC1 and PC2 on the prototype network shown in Figure 3 - 5. The Dynamic.xsl file is analysed adding the correction factor to the intrinsic

score as the different security organisations are given the same vulnerability, different levels of severity. The intrinsic correction factors are used to evaluate the intrinsic scoring, the corrections.xsl file has been used with a mathematical formula to calculate the correction factor, which might have negative values to adjust the intrinsic scoring.

A new method to represent a unique severity dynamic cost of the total weight of all vulnerabilities is developed for each host. We calculate the severity cost for all vulnerabilities based on the DVSS score and then process the existing results by classifying them to none, user and root privilege levels. We may need to tolerate some of the vulnerabilities, and to take countermeasures only for critical vulnerabilities when a multi-step intrusion is actually happening. Accordingly, within each level, each of them classified to operational level 0, 1 and 2 (a multi-step intrusion) severity cost and then use the existing results to build the cost-centric attack graphs.

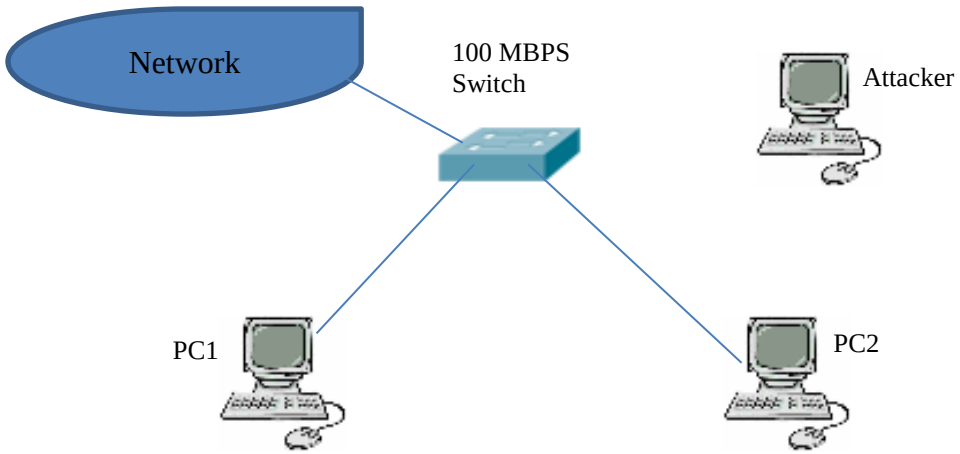


Figure 3-5. Using Nessus scanner as a tool to discover known Vulnerabilities on PC1 and PC2

The vulnerabilities classified into three levels, according to privileges (user, root/administrator, none). The user privileges are presented, those, which the intruder could exploit, the vulnerabilities with user account privilege or gain a user account privilege. While root privileges level are refers to the vulnerabilities that

could give the intruder root or admin privilege after exploiting the vulnerabilities and none privileges are presented the attackers, which can exploit the vulnerability from the external networks or website with no privilege escalations. User, root, and none privileges are shown in Table 3-2.

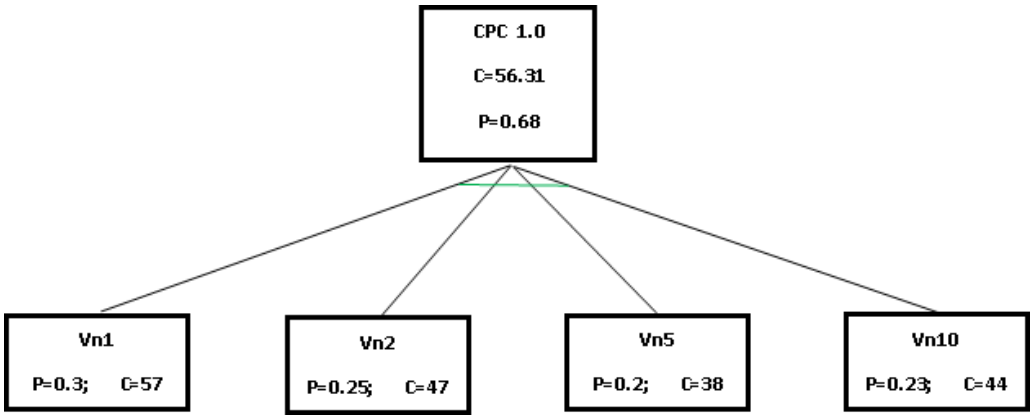


Figure 3-6. Using AT structure to calculate OLC,

In this model, AT is represented using cost centric driven from DVSS as a new approach to building effective AT. Exploitation of more than one vulnerability in one asset is possible in terms of AND and OR refinement associations between vulnerabilities in different time slots. For calculating the effective cost, according to operational levels, only, one vulnerability from an operational level group could be exploited at a given time. Based on this fact the relationships between vulnerabilities in terms of calculating the effective cost for each operational level will use only OR refinement to adjust the operational costs to be near to the higher cost vulnerability in the group. To calculate the OLC CPC 1, 0, which means the average severity of PC-1 vulnerabilities with none privileges, we will use the following formula and methods:

As shown in Figure 3-6 and Table 3-2, the OLC- CPC1, 0 for Vn1, Vn2, Vn5 and Vn10 vulnerabilities calculated as follows:

$$P_{PC1,0} = 1 - (\prod_{k=1}^{n} (1 - P_k)$$

$$P_{PC1,0} = 1\text{-}((1\text{-}0.31)*(1\text{-}0.25)*(1\text{-}0.20)*(1\text{-}0.\ 0.24)))$$

$$CPC1,0 = Max(C) - ((\sum_{k=1}^{n-1} p_k * C_k) / (\sum_{k=1}^{n} p_k /100))$$

$$CPC1,0 = 57 - (((47*0.25) + (37*0.20) + (44*0.24))/100)/((0.25 + 0.20 + 0.25))$$

$$CPC1,0 = 56.31.$$

Using AT to calculate the OLC for CPC 1, 0, CPC 1.1 and CPC 1, 2 shown in Table 3-2 and Figure 3-7.

	Vn1	Vn2	Vn5	Vn10			OLC	Value
Probability	**0.31**	**0.25**	**0.20**	**0.24**				
None privileges	**57**	**47**	**38**	**44**			**CPC1,0**	**56.31**
	Vn4	**Vn6**	**Vn9**	**Vn11**				
Probability	**0.23**	**0.31**	**0.26**	**0.19**				
User privileges	**52**	**71**	**60**	**44**			**CPC1,1**	**70.47**
	Vn3	**Vn7**	**Vn8**					
Probability	**0.33**	**0.33**	**0.33**					
Root/Administrator privilege	**89**	**90**	**87**				**CPC1,2**	**89.12**

Table 3-2. Operational levels of PC1 Host

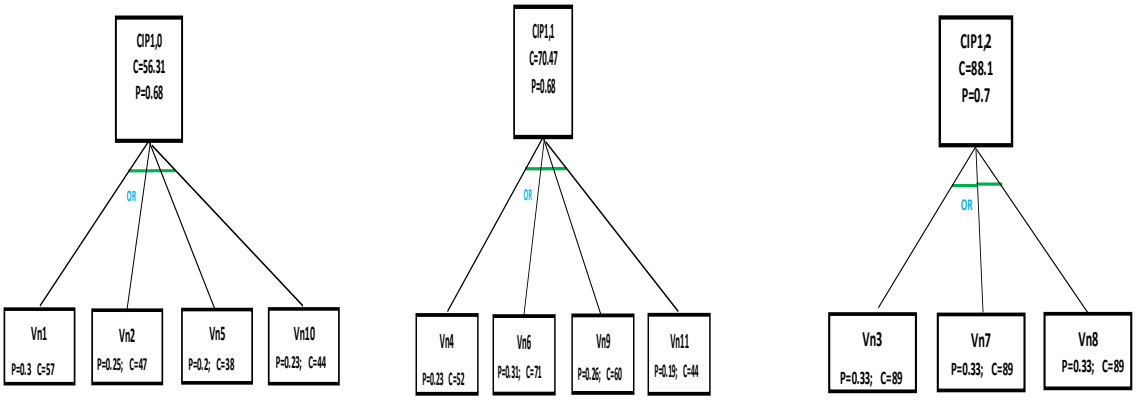
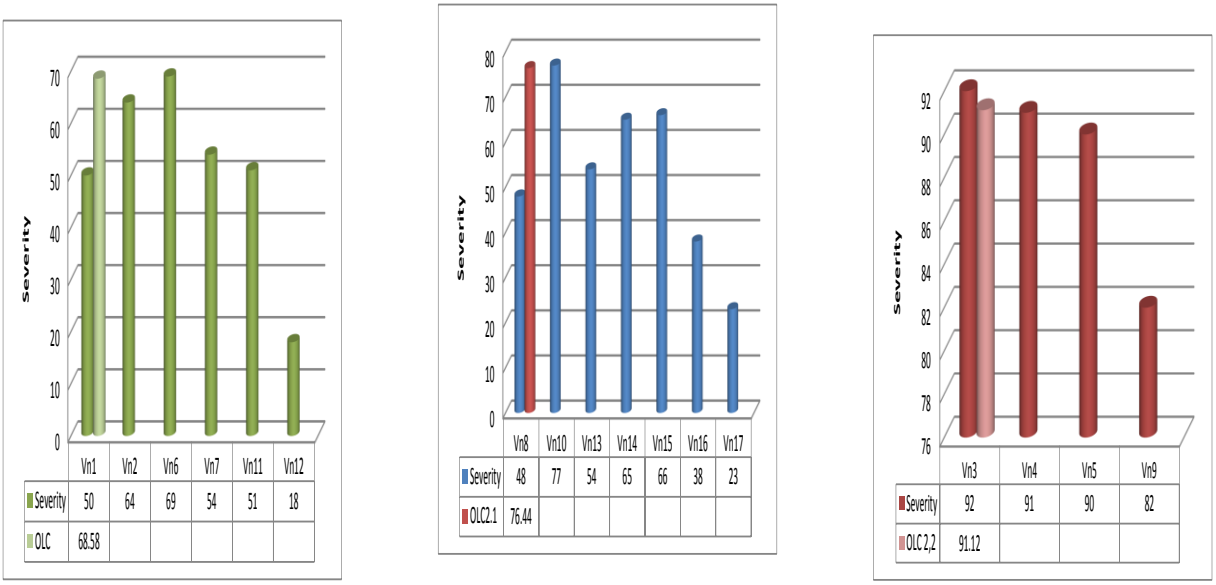


Figure 3-7. Vulnerabilities distribution for PC1

In the same manner, the average level of vulnerabilities is calculated for OLCs as shown in Figure 3- 8 and table 3- 3.

	Vn1	Vn2	Vn6	Vn7	Vn11	Vn12			OLC	Value
Probability	0.16	0.21	0.23	0.18	0.17	0.06				
None privileges	50	64	69	54	51	18			CPC2,0	68.58
	Vn8	Vn10	Vn13	Vn14	Vn15	Vn16	Vn17			
Probability	0.13	0.21	0.15	0.18	0.18	0.10	0.06			
User privileges	48	77	54	65	66	38	23		CPC2,1	76.47
	Vn3	Vn4	Vn5	Vn9						
Probability	0.26	0.26	0.25	0.23						
Root/Administrator privilege	92	91	90	82					CPC2,2	91.12

Table 3-3. Operational levels of PC2 Host



CPC1, 0` CPC1, 1 CPC1, 2

Figure 3-8. Vulnerabilities distribution for PC2

In the OLC model, the attributes of hosts PC1, PC2 include:

1- Intruder's right of entry privilege for each host: the privilege level, classified as system administrator/ root (2), regular user (1), guest or none (0).

2- Security Metrics cost for each host: represents the CVSS score with the Risk Assessment levels for each host.

3- Exploit mode: The attacker location (e.g. Local, Network) to perform an attack/exploit.

3.9 RESULTS AND OUTCOMES OF FRAMEWORK TESTING

In practical testing of the framework demonstrated in section 3.7, the cost-centric approach was tested using a prototype network. The Nessus Scanner 4.x, developed open source program and a VLAN is used to separate the ports in the Ethernet switch and measures the influence on the dynamic.xsl file and the CVSS calculator demonstrated in the Appendix- is used to benchmark the DVSS scores with CVSS to monitor the impact of using a dynamic approach of calculation vulnerabilities scoring. The method of vulnerability classification is divided into three levels, according to privileges (user, root/administrator, none) is tested by comparing the most critical vulnerability scoring. To make sure that the calculated values are reflect a real representation of the vulnerabilities existing in the hosts and the operational levels of the vulnerabilities states for each privileges are rigor and significant and it will lead to correct results, at the time when the dynamic cost-centric values used to builds and ranks the attack graph.

The following results are presented according to the test plan demonstrated in section 3.7:

1- As shown in Figure 3-8 and Table 3-3, the dynamic severity costs driven from this approach according to DVSS scoring in each host, using OLCs are comparable to each other (DVSS and CVSS scoring values). If the dynamic impact and correction value are considered to the intrinsic scoring that will prove the framework used in this test is capable of providing an accurate

dynamic cost-centric metric for each state of the host and classify and calculate the dynamic vulnerability costs according to privileges.

2- A VLAN is used to separate the ports in the Ethernet switch and measures the influence on the dynamic.xsl file. The CVSS framework demonstrated in the Appendix-B is used to benchmarking the DVSS scores with CVSS, to monitor the impact of using a dynamic approach in the calculation of vulnerabilities scoring. In section 3.4, which demonstrate the significant and accurateness of the dynamic cost approach use in the framework to precede the analysis of the vulnerability attributes.

3- Using a different platform, Windows and UNIX in the practical example, with Nessus scanners and using dynamic files and the program proved that the proposed method produces a security standard solution and met the system specification.

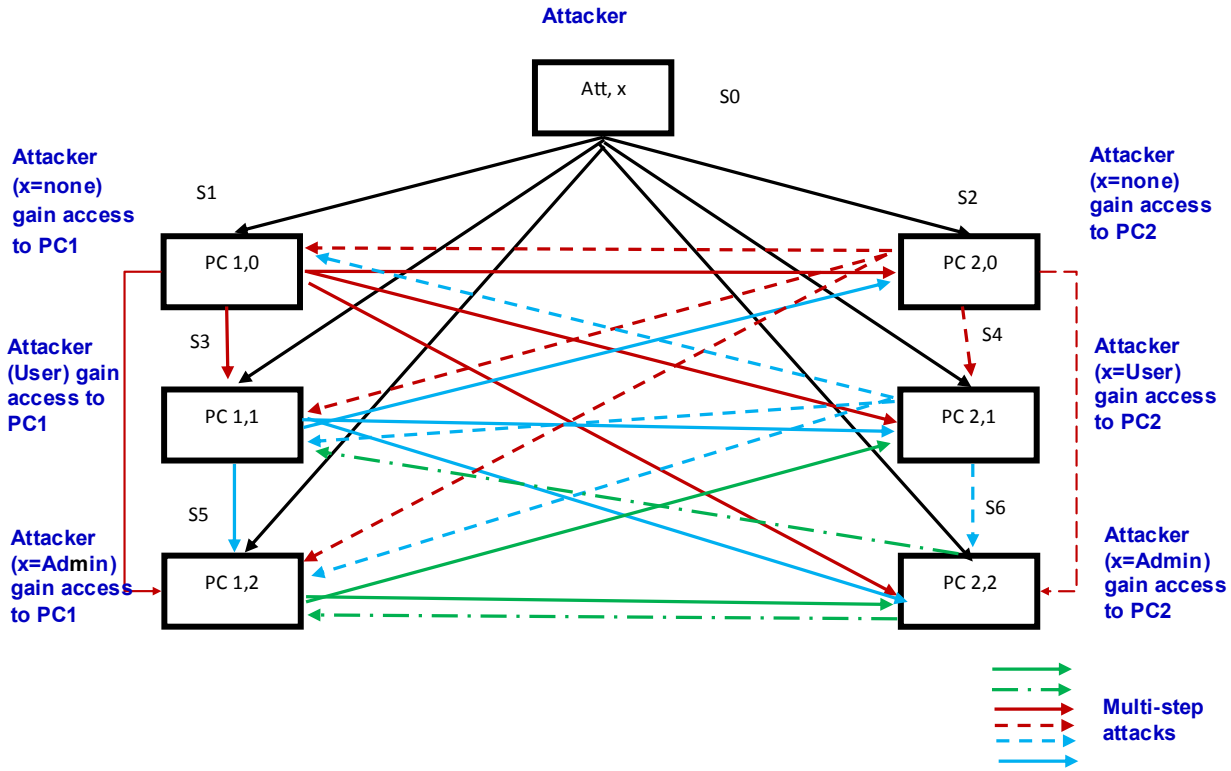
4- The AT used in this approach and lead to proper and accurate results, however, using this structure need a lot of adjustments and collaborations as in some cases the flow of the costs and probabilities could lead to improper results. The same issues are occurring with the hardware and software setup, for example, incorrect reading of the information from dynamic.xsl file leads to some mistakes in the dynamic calculations even when the manual calculations give a correct result due to some problems in digital communications. Which take some time for fixing them to get the correct results that indicate the need to improve the quality of the programs and tools were used in the investigations.

5- As you can see from Figure 3- 9, AG is using dynamic impact cost-centric metrics, with a total of 7 states to represent the dynamic cost centric attack states for PC1 and PC2 and all possible paths. The main goal is to reduce the visual complexity of an AG using a CostRank methodology (as we will explain in the next chapter); the presentation of the full-scale scenario for the test pad indicates a dramatic reduction of the complexity of an AG. State zero (s0) represents the attacker initial stage, an assumption has been made that the attacker can initiate the attack on his/her own PC and he has full

privileges on it (ATT, 2). The other state represents the status of PC1 and PC2 according to the vulnerabilities type. Exploit S1 means the attacker exploits a vulnerability in PC1 without gaining any privileges and without any authentications (PC 1, 0). While S3 means the attacker exploits a vulnerability with a user's privilege or the attacker escalates the privileges as user as a result of the attack. S5 allows the attackers to gain root/administrator privileges because of exploiting a vulnerability.
Please notice the direct cost to reach S1, S2, S3, S4, S5, and S6 represent OLC values shows below and in table 3-2 and 3-3.

CPC 1, 0 = 56.31 = (S0, S1)
CPC 1, 1 = 70.47 = (S0, S3)
CPC 1, 2 = 89.12= (S0, S5)
CPC 2, 0 = 68.58= (S0, S2)
CPC 2, 1 =76.47= (S0, S4)
CPC 2, 2 =91.12= (S0, S6)

The indirect cost in multi-step attack will depend mainly on the destination state as shown in Figure 3-9. For example (S3, S2) cost=68.58 as the attacker tries to exploit the vulnerability in PC2 with OLC 2, 0.



S0	S1	S2	S3	S4	S5	S6
Attacker	PC 1, 0	PC 2, 0	PC 1, 1	PC 2, 1	PC 1, 2	PC 2, 2
	56.31	68.58	70.47	76.47	89.12	91.12

Figure 3-9. Dynamic Cost centric attack graph (a full-scale scenario)

As the main goal of this thesis, to reduce the visual complexity of the attack graph, the benchmarking of the dynamic cost-centric AG approach with the classical method is substantial to illustrate the effectiveness and significance of the method. Figure 3-10, shows a) Prototype network for experiments taken from [Reginald Sawilla, 2008] and b) corresponding attack graph. The assailant, in this experiment has a direct access to PC1, which causes just one vulnerability Vu1 and PC2, which has likewise only one vulnerability Vu2. PC1 and PC2 have access to the destination machine, which possess just one vulnerability Vu3, there are no connection between PC1 and PC2.

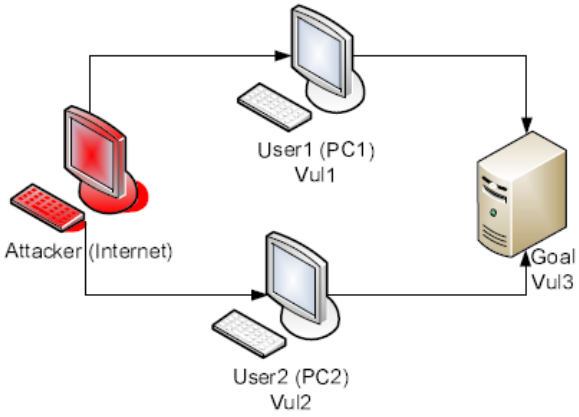
In this experiment, the authors generate the AG only for three vulnerabilities as shown in Figure 3-10 (b), the vertex colours used to represent the vulnerabilities with low and high risk, representing from blue to red.

In the experiment using dynamic cost-centric AG, there are **10** vulnerabilities in PC1 and 17 vulnerabilities in PC2, the AG representation in Figure 3-9, represents a full-scale scenario, taking into account all the possibilities of multi-step attack. Just as, you can observe, the vulnerabilities in the instance represented in Figure 3-9 are increased approximately 6 times.

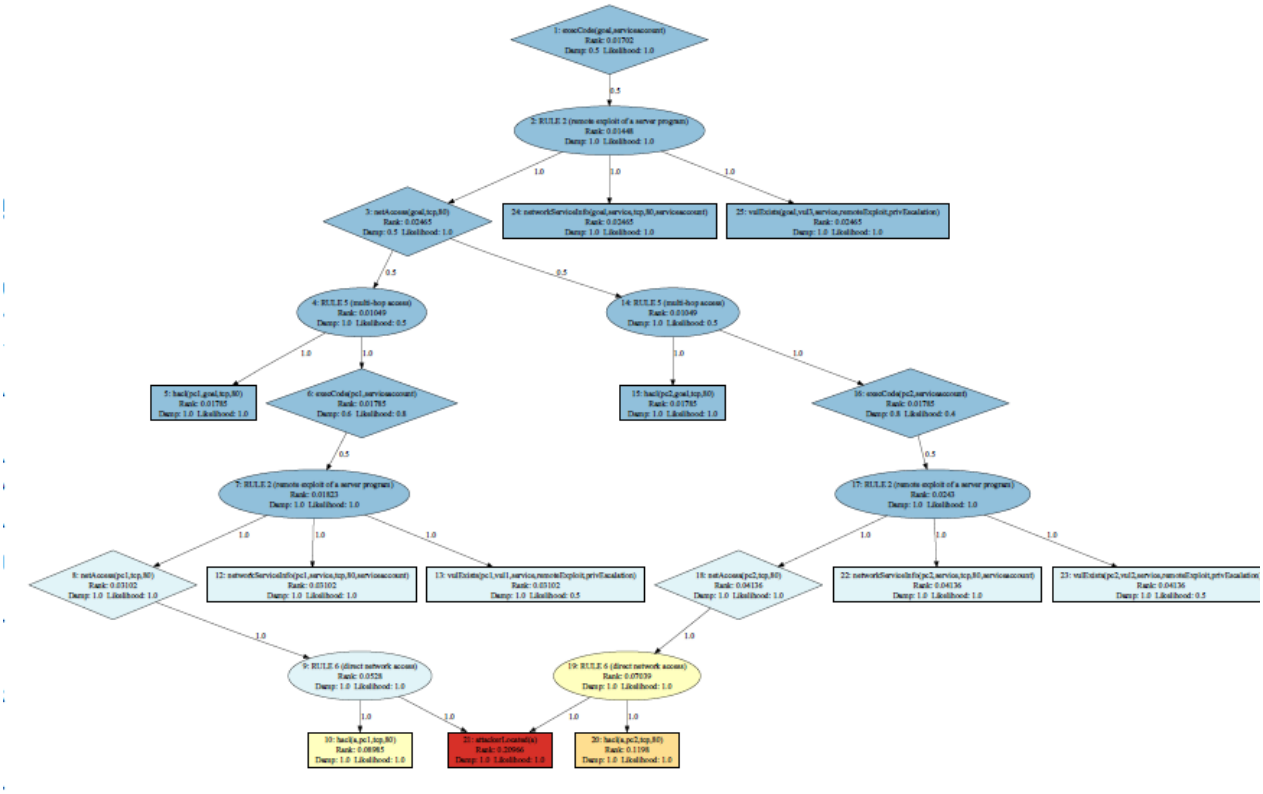
Using a full-scale scenario, the AG is much less complicated and the information presented is more comprehensive for system administrators and security professional.

When the ranking algorithm will be utilised to determine the peak serious states (in Chapter-4), then an effective cost-centric approach can be applied to protect a critical asset and to carry out an effective countermeasure as the following chapters will show.

As an outcome, a novel and significant approach are presented, to develop a dynamic cost-centric scoring system using OLC, producing a unique severity costs for each state of the host. The cost-centric values are used to build a novel cost-centric AG to reduce the visual complexity and to assist the security professionals and system administrators in security decisions and to create and maintain security policies and procedures.



a) Prototype network experiment



b) A classical attack graph of prototype network (a).

Figure 3-10. A classical attack graph of the Prototype network (a) taken from [Reginald Sawilla, 2008]

3.10 CONCLUSION

This chapter, described a novel approaches, for developing a tool for dynamic vulnerability quantitative scoring. Including network topological analysis to measure the impact of network devices and configurations and considering the correction factor to change the scores of the intrinsic metric (as the different security organisations are given to the same vulnerability different levels of severity). In addition, the influence of these factors on the final scoring value of the vulnerabilities, as the network topology and configuration change, the formulae takes different scoring values from the developed dynamic database file instead of constant empirical static values used by CVSS.

The dynamic impact scoring of the vulnerabilities is then used, to advance the cost-centric method to assign a unique severity cost for each host. By dividing the scores of the vulnerabilities to three main levels of privileges: (i) none; (ii) user and (iii) root, and then classified these levels into operational levels to identify and calculate the severity cost of multi-step vulnerabilities. These advances and methods that provides an important stride towards reaching the objective of securing network systems and to assist the security investment decision makers in the process of choosing a proper security solution based on the cost/benefit analysis, as the cost-centric method will be utilised in building and placing a novel cost-centric attack graph.

A unique dynamic severity cost model using the total weight of all vulnerabilities for each host has been developed, using DVSS scores of intrinsic, time-based, and ecological metrics by combining related sub-scores and modelling the problem's parameters using mathematical framework. In the test pad, Nessus Scanner 4.x was used to discover known vulnerabilities, an open source program has been modified to process and combine different dynamic files to produce dynamic scores. The CVSS Framework demonstrated in the Appendix-B is used to benchmark the DVSS scores with CVSS to monitor the impact of using a dynamic approach of calculation vulnerabilities scoring.

As the risk should be evaluated in terms of maximum impact on an adverse event, the risk matrix is used and modified formulae to collaborate the risk equations.

The method of vulnerability classification into three levels, according to privileges (user, root/administrator, none) is tested. By comparison, the most critical vulnerability scoring to make sure that the calculated values reflect a real representation of the vulnerabilities existing in the hosts. In addition, the operational levels of the vulnerabilities states for each privileges are rigor and significant and it will lead to correct results, at the time when the dynamic cost-centric values are used to build and rank the attack graph. Using a cost - centric framework to classify the vulnerabilities to none, user and root privileges, and then finding the scores to OLC, the practical approach, demonstrated that this classification really reflects the use of the risk matrices. The cost-centric values are used to build a novel DCCAG to reduce the visual complexity and to assist the security professionals and system administrators in security decisions and to create and maintain security policies and procedures. An example of a dynamic Cost centric attack graph is presented with unique dynamic cost assignments according to different operational levels calculated using the cost-centric framework and formulae.

In the next chapter, a novel approach to building up a scalable CostRank algorithm for ranking the AG using a new approach to represent the states by implementing a dynamic cost-centric driven from statistical probability of dynamic impact scoring for the vulnerabilities.

The ranks of the states represent the importance of the states in the Dynamic Cost-Centric Attack Graph (DCCAT). The resulting ranking represents a metric that can be used by security professionals and organisation administrators to establish different security decisions in order to improve the network protection based on the financial cost effective approach. As CCAG is represents the key step to elaborate the dynamic security business case.

CHAPTER 4.COST-CENTRIC ATTACK GRAPHS BASED ON DYNAMIC SECURITY METRICS ANALYSIS

4.1 INTRODUCTION

.

As the number of vulnerabilities and hosts in the network rises, the process of evaluating the critical vulnerabilities, which can be used by attackers, turns out to be vital. In modern computers, networks, there are usually several platforms, many software application packages, and different network technologies, so it is not sufficient to consider individual (static) vulnerabilities to evaluate security because using these approaches will lead to many security holes that most security professionals will be unable to discover.

The attack graph (AG) is a security model representing the chains of vulnerability exploits in a network. A number of researchers have acknowledged the attack graph visual complexity and the lack of in-depth understanding.

Each Node in the AG represents a single attacker state (action), the paths in the AG represent a sequence of states that an attacker can use in a multi-step attack to exploit a vulnerability in the target's potential assets. This results in a scenario where the attacker has breached network security.

Classical AG methods evaluate the protection required by bringing into account the vulnerabilities in each host by using vulnerability-scanning tools.

The interactions between the individual vulnerabilities and the network architectures and configurations are not considered in most AG models.

In the dynamic cost-centric AG, the paths of an AG are represents a chain of the states of dynamic vulnerabilities. These are referred to as states, where each state signifies an action of an attacker, such as the attacker gaining administrator/Root privileges on a specific host.

In this novel approach, statistical probability mitigation to dynamic vulnerability impact model is used to construct a dynamic metrics attack graph.
The network architectures and configurations, along with the interactions between the individual vulnerabilities are taken during the calculations of the cost. These are exemplified by the Dynamic Vulnerability Scoring System (DVSS) and an associated the dynamic cost framework.
Migrating statistical probability of the dynamic impact model is the key to this thesis, as in objective 1, 2 are achieved, referring to Chapter-3, A new methodology was built up to present an attack graph with a dynamic cost metric and also a novel methodology to estimate and represent the cost for each vulnerability states was developed.
The next phase, as demonstrated in objective-3, is to develop a ranking method for cost-centric attack graphs to minimise dimension and complexity of the attack graphs.

As the classical AG suffers from visual complexity, the need of ranking techniques has arisen when performing link analysis to deal with these problems, the author studied some poplar ranking algorithms to develop a ranking method for CCAG. PageRank [Arasu, Arvind, et al., 2002] is the most respected ranking algorithms, considered equally a narrative of success for Google. Other ranking algorithms are considered to be less popular, such as, for example, Hyperlink Induced Topic Search (HITS) [Edge, Kenneth, 2007], Hub-Rank, Sym-Rank and Auth-Rank. All of these methods use Stochastic techniques for link analysis. Recently, researchers [Kijsanayothin, Phongphun, 2010] suggested the idea of using a modified PageRank algorithm in dependency AG to rank the graph and detect the most effective vulnerabilities for proper analysis and to find effective countermeasures and mitigations.
In this Novel approach, CostRank algorithm was developed (based on PageRank) using stochastic Markov model calculations taking into account the dynamic cost-centric approach via matrix's Eigenvalue calculations to rank the states of vulnerabilities using statistical probability of the dynamic vulnerability impact model. This approach enhanced the classical approaches of constructing

and analysis of an AG. As the CCAG is based on dynamic vulnerability quantitative scoring, including network topological analysis to measure the impact of network devices and configurations. Considering the correction factor to change the scores of Intrinsic metric of DVSS and the interactions between vulnerabilities as discussed in Chapter-3, which reduced the problem of visual complexity in classical AG approaches, which represents a chain of individual (static) vulnerabilities. In this approach, the states of the host are represented instead of each vulnerability, using the operational levels to model the cost of the states.

Analysis and comparison have been conducted on the existing results of the ranking algorithms with those of Mehta et al. and Kijsanayothin approaches respectively, as these particular approaches are highly valued and published in high-ranking journals in the field of AG. The same undertaking was applied to a prototype network as a proof-of-concept validation of the model. Additionally, graphs and modelling environments have been extended in order to facilitate the computation of more complex metrics. The existing results demonstrate the effects of the algorithm. The dynamic cost-centric AG can be used as valuable tools in forensic investigation, detection and mitigation of the critical vulnerabilities for analysis and mitigations intrusion detection and defence.

There are four types of graph [Manaskasemsak, Bundit, and Arnon Rungsawang., 2005] [Sawilla, Reginald Elias, 2011]:

i. Undirected graphs
ii. Directed graphs
iii. Undirected hypergraphs
iv. Directed hypergraphs

Detailed information about these four types is shown below with examples of suitability of data representations for each type:

4.1.1 Undirected graphs

An undirected graph could be defined in terms of the relationship between vertices, as the relation between any two vertices is symmetric, (the edge not assigned with the direction attributes as in a directed graph).

U_G represents the undirected graph, $U_G = (V_x, E)$ which has a set of vertex Vx and a set of edges E that connect different vertices to each other. The edge set features could be represented as follows:

$E_U = \{(v_n, v_m) \mid v_n, v_m \in V_x\}$ where n, m represents the vertex's number, the above sets containing two vertices indicating that an edge occurs between them.
The representation of the data in Facebook or Twitter or any other social network is more suitable to an undirected graph. In Facebook, for example, the vertex symbolises the individuals and the edges represent the connection (friendships) between them.

To justify the suitability of data representations with an undirected graph:
Since the relationships between friends are usually both one to one and bi-directional, an undirected graph is most suitable.

Figure 4.1 shows an example of an undirected graph, the vertices represented from vertex-1 to 6.

$$V_U = \{v_1, v_2, v_3, v_4, v_5, v_6\}$$

The set of edges represents the relationships between vertices as follows:

$$E_U = \{(v_1, v_2), (v_1, v_3), (v_1, v_4), (v_1, v_6), (v_2, v_3), (v_3, v_5), (v_4, v_5)\}.$$

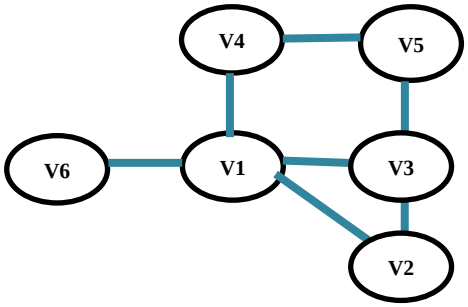


Figure 4-1. An example of an undirected graph modified from [Sawilla, Reginald Elias, 2011]

4.1.2 Directed acyclic graphs

In directed Graphs the relationship between vertices are Asymmetric, (the edge assigns with the direction attribute). The basic/simple directed graph is a representation in the form of a graph, but without loops (cycles) and with no compound/several arcs/ edges. The number of simple directed graphs of n nodes for n=1, 2 ...x are 1, 3, 16, 218, 9608 …

The graph is called a directed graph if the graph has directed arcs (edges) such as Ed $=\rightarrow (v_n, v_m)$ is called directed graph from v_n to v_m where n, m is the edge numbers, v_m represents the head and v_n represents the tail. The number of heads brings up to a specific node called <u>in-degree</u> and amount of tails emerge from a specific node called <u>out-degree</u>, the edge set defined as:

$$E_D = \{\rightarrow (v_n, v_m) \mid v_n, v_m \in V\}$$

A web graph that contains websites links to each other is an example of data suitable to the directed graph. Each vertex represents a website and the edges represent the links (hyperlinks) from one website to another. The link between webpage X to webpage Y does not indicate there is a reverse link between webpage Y to webpage X, that justifies the web graph data is suitable to direct graphs. Figure 4.2 shows direct graphs representing a web graph.

The web pages represented as graph vertex as follows:

$$V_D = \{v_1, v_2, v_3, v_4, v_5, v_6\}$$

The set of edges (arcs) is represented as follows:

$$E_D = \{\rightarrow(v_1, v_3), \rightarrow(v_2, v_3), \rightarrow(v_3, v_4), \rightarrow(v_3,v_5), \rightarrow(v_3,v_6), \rightarrow(v_4,v_5), \rightarrow(v_6,v_3)\}$$

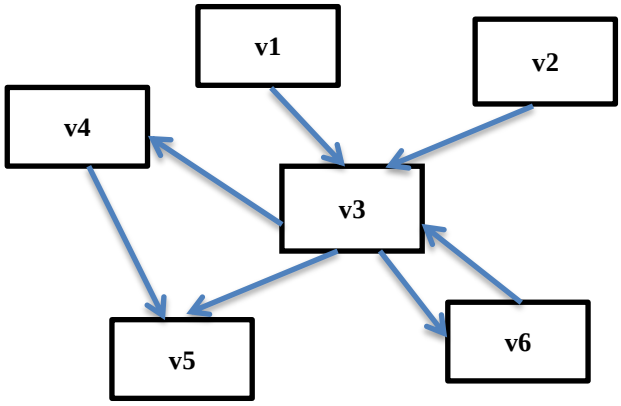


Figure 4-2. Illustration of a directed graph modified from [Sawilla, Reginald Elias, 2011]

4.1.3 Undirected Hyper graphs

A hyper graph [Sawilla, Reginald Elias, 2011] is an overview of a graph where arcs/edges could bond any number of vertices.

An undirected hyper graph satisfies the condition that each edge be comprised of exactly two vertexes. These arrangements will not be used to characterise other dense sets of data apart from undirected graphs. While they do not represent more complicated sets of data than undirected graphs, if the data set is naturally represented in 1-2-1 (one to one) fashion, the undirected hyper graphs can present the data in a more squeezed style [Aykanat, Cevdet et al.,2008] [Sawilla, Reginald Elias, 2011].

The hype edge could be defined as a single undirected edge. The hyper graph can attach several vertices using a hyper edge. A hyper graph could be represented as:

$$G_{HU} = (V_H, E_H)$$

As shown above, the hyper graph encompasses a group of vertices V_H with a group of edges E_H which connects the vertices. A facet $e_n \in E_H$ (where n represents the edge number) is a set $e_n = \{v_1, v_2, \ldots, v_k\}$ denoting that all of the vertices in e_n are connected by a particular edge n. The hyper graphs can be described as S-Uniform if all the edges connect exact S vertices. Using the same logic, the undirected graphs could be called 2-uniform.

An example of data's suitability to the hyper graph representation is a group of students and the courses they study. Figure 4.3 shows a hyper graph of students and courses, students represented by vertices, and the courses they study represented by hyper edges. Students are represented by the set of vertices as follows

$$V_H = \{v_1, v_2, v_3, v_4, v_5\}$$

The courses are symbolised by groups of hyper edges

$$E_H = \{\{v_1, v_2\}, \{v_1, v_2, v_3, v_5\}, \{v_5, v_4\}\}.$$

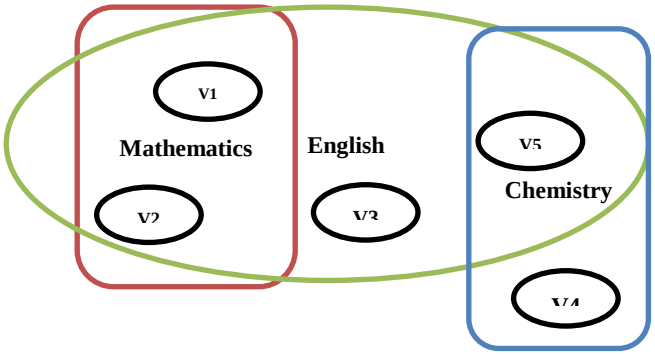


Figure 4-3. An instance of an undirected hyper graph modified from [Sawilla, Reginald Elias, 2011]

4.1.4 Directed Hyper graphs

A directed hyper graph is a hyper graph with direct hyper edges. Directed hyper graphs are the common generalisation of both directed graphs and undirected hyper graphs. A hyper graph could be represented as:

$$G_{DH} = (V_D, E_{HD})$$

The directed hyper graph consists of vertices V_D and an E_D group of direct hyper edges or hyper arcs.

A facet $e_n \in E_{HD}$ (where n represents the edge number) is a set $e_n = \{v_1, v_2, \ldots, v_k\}$ denoting that all of the vertices in e_n are connected by a particular edge, n is an ordered pair of detached vertex group $e_n = (V_D^{-}{}_{\rightarrow}, V_D^{+}{}_{\leftarrow})$. Where the vertices in the group $V^{-}(\rightarrow)$ are the resources (Tail) of the directed hyper edge and the vertices in the set $V^{+}(\leftarrow)$ are the targets (Head) [Sawilla, Reginald Elias, 2011] Specific rates/weights (ς) are assigned to the edges using the following formula:

$$Cost : V_D x E_{HD} \rightarrow \varsigma$$

ς will represent negative real numbers for source vertices and positive real numbers for the target vertices of a specific edge.

An F_w-arc (forward hyperarc) is a directed hyper edge where cardinality of a source vertex set is one, $|V^{+\leftarrow}| = |\varsigma| = 1$, a B_w-arc (backward hyperarc) is a directed hyper edge where cardinality of source vertex set is one, $|V^{-\rightarrow}| = |\varsigma| = -1$. An F_w-graph (forward hyper graph) is a directed hyper graph whose edges are F_w-arcs, a B_w-graph (backward hyper graph) is a directed hyper graph whose edges are B_w-arcs.

An example of data's suitability to the direct hyper graph representation is an exploit against computer network vulnerabilities. The vertices in the directed hyper graph represent the attacker privileges; Fw Arcs represent the preconditions that enable the attacker to gain new privileges during an attack.

An example of directed hyper graph represented in Figure 4.4, where the vertices symbolise the attacker privileges as follows:

$$V_D = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$$

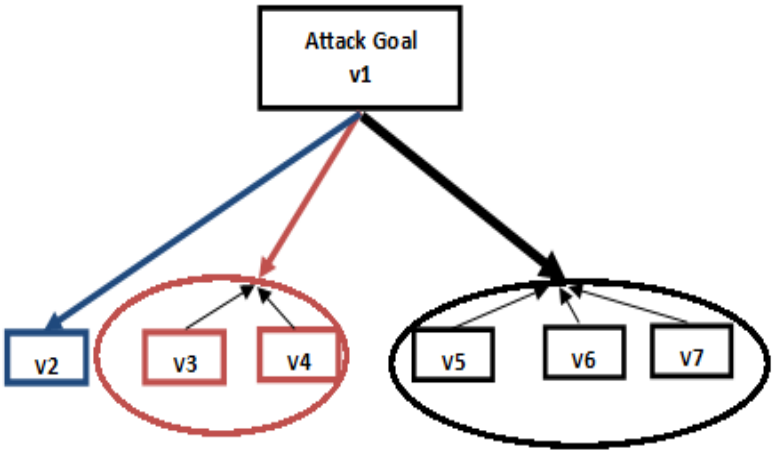


Figure 4-4. Model of a directed hyper graph

The preconditions to satisfy the goal target v1 are represented in terms of the following vertices:

{v2} or {v3, v4} or {v5, v6, v7}.

$$E_{HD} = \{((v_1) \leftarrow (v_2)), ((v_1) \leftarrow (v_3, v_4)), ((v_1) \leftarrow (v_5, v_6, v_7))\}.$$

4.2 ATTACK GRAPHS CLASSIFICATIONS

Much research in network security has focused on generating attack graphs from attack paths composed of states and vulnerability exploits, according to a probable grouping. The attack graph could be organised [Steven Templeton and Karl Levitt, 2000] as:

1. State oriented Attack Graph.
2. Exploit oriented Attack Graph.
3. State Exploit oriented Attack Graph.

4.2.1 State-Oriented Attack Graph

In this class, the vertices represent a group of the network states and the attacks/exploits represent the edges to show the transitions from one state to a different state in the network.

The attack/exploit is an utilisation of vulnerability; the descriptions of an exploit will be in terms of its preconditions and impacts. The characteristic of the network or group of network characteristics can describe the term network state, such as network node/host, node connectivity, and installed/upgraded/modified software application on a specific node, privilege access right or any other character such as privilege level.

The vulnerabilities are described in terms of their preconditions and post conditions in the state oriented attack graphs as shown in Figure 4-5 taken from [Bhattacharya, Somak, and S. K. Ghosh, 2008], the attacker can grant more states in the network if the preconditions of the vulnerabilities are fulfilled.

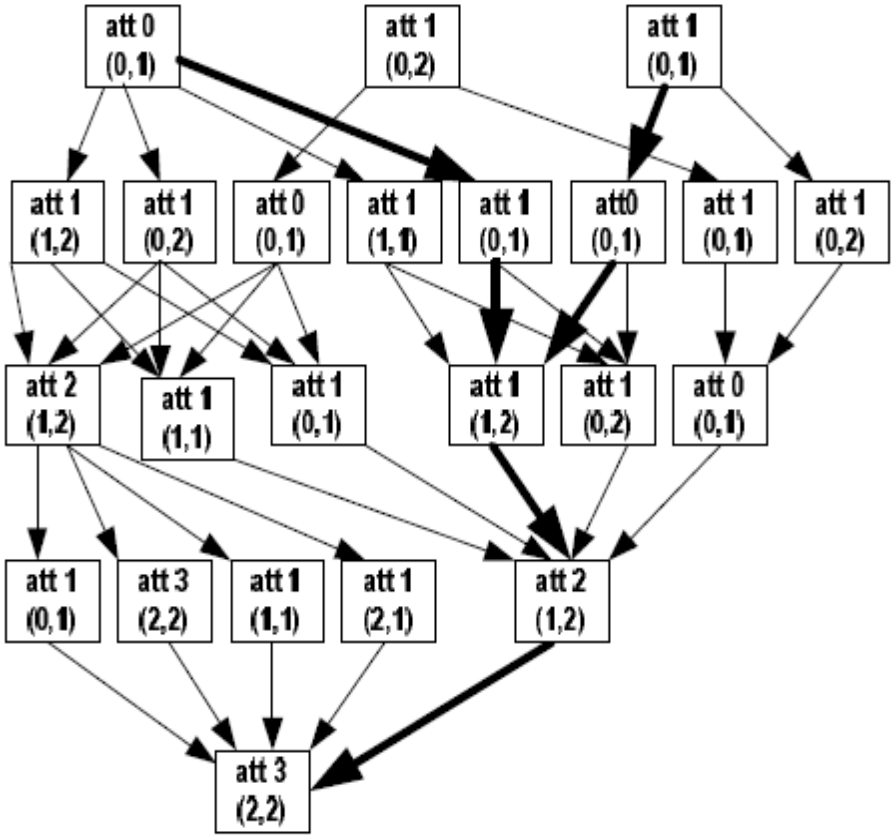


Figure 4-5. State-Oriented attack graph generated by model checker [Bhattacharya, Somak, and S. K. Ghosh., 2008].

Referring to Figure 4.5, which represents a state oriented attack graph; in this example, there are two incidents of two attacks.

The ftp bounce attack (att 0) is an attack/exploit of ftp software vulnerability; in this exploit the attackers using “port” command to get access to the ports on the target host/node using a specific host as middle host/man.

The secure shell (ssh) buffer overflow (att1) represents vulnerability in the software applications (web site) by sending very large packets and executing an arbitrary code to get access to the remote host through a network to copy, modify, delete or add files or change configurations and settings.

These two exploits att0, att1 are not related to each other in their precondition or their post condition in the network specifications.

The attacks can be moved in any direction, for example:

att 1 (0, 2) → att 1 (0, 1) → att 0 (1, 1) → att 2 (1, 2) → att 3 (2 , 2)

or

att 0 (0, 1) → att 1 (1, 1) → att 1 (1, 2) → att 2 (1, 2) → att 3 (2, 2)

Each vertex in the state oriented attack graph represents a state assigned with a digital number 0, 1, 2, 3,... n which represents node-1, node-2, node-3, ...node-n.

The main goal of a state oriented attack graph model is to discover possible paths from the source host (attacker host) to the target host in terms of exploiting vulnerabilities and gaining privileged access, in efficient manners, acceptable timing and finding appropriate countermeasures to mitigate vulnerabilities across the paths.

The model should contain the appropriate tools to represent all states of the network in terms of connectivity, privileges, configurations, vulnerabilities and other recourses in order to represent different paths the attackers might use to exploit vulnerabilities. The representation of a different state in the following methods for example

User > Administrator>S

Mean the user on host S has administrator privilege.

Buffer overflow > N

FTP > H1

This will return true/false for the first one; it means, the buffer overflow can run and execute arbitrary code to gain privileges on host N and the second one means the ftp services can run on host H1.

4.2.2 Exploit dependency/Oriented Attack Graph

In the exploit oriented attack graph, the vertices signify conditions and the edges represent the exploits. This is the converse of the state oriented attack graph and sometimes, refers to it as an exploit dependency, attack graph.

The representation of exploiting oriented attack graphs comes with primary condition/s and goal condition/s and the states of some unique hosts. The exploit-oriented attack graph's initial state(s) and the goal state(s) of the network are special nodes. Primary condition/s symbolises the exploit hosts with exact post conditions and void/null preconditions. Goal condition/s represents the exploit hosts with exact preconditions and void/null post conditions.

Figure 4.6 demonstrates the exploit oriented attack graphs; the oval symbols represent the exploits and the plain text represents the conditions, as exceptions the triple octagon represents the goal conditions[Steven Templeton and Karl Levitt., 2000].

For example:

Ftp (1, 2) → trust (2, 0) → user (2) → trust (1, 2)

The numerical values in the parenthesis represent specific hosts.
FTP (1, 2) indicates execute FTP on host 1 to transfer files from host 2, user (2) indicates using user privilege on host 2 and trust (1, 2) indicate host 1 trust host 2.

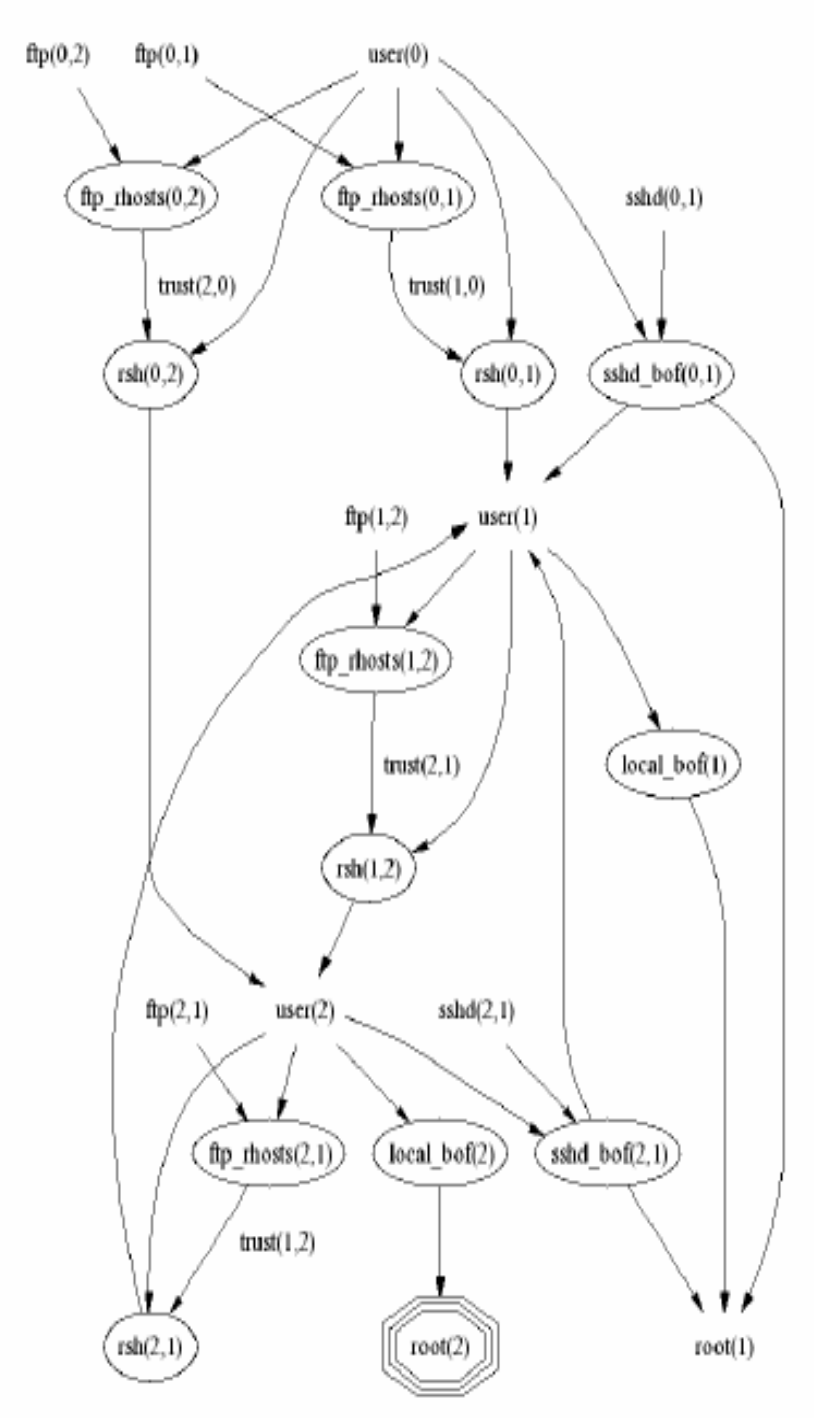


Figure 4-6. Exploit dependency/oriented attack graph [Noel, Steven, et al., 2003]

4.2.3 State-Exploit-Oriented attack graph

The states and exploits are represented as vertices in the state-exploit oriented attack graph [Steven Templeton and Karl Levitt, 2000]. An edge in this graph could relate to a specific state, to a specific exploit or a specific exploit and a state. The edge will not directly relate a state to another state or an exploit to another exploit.

In this thesis, a state oriented attack graph is used. As a state is a network attribute or set of network attributes, network state, such as network node/host, node connectivity, and installed /upgraded/modified software application on a specific node, privilege access right or any other render character such as privilege level. The goal is to develop a DCCAG approach where states are used to describe vulnerabilities, and the state-oriented attack graph model has been modified to represent the dynamic cost metric approach methodologies. The state oriented attack graph was more appropriate in this research, as the cost-centric framework is used to represent the dynamic state for the vulnerabilities existing is a specific critical asset (host) according to the different operational levels based on privilege, the cost of exploiting a vulnerability is determined according the accumulation approach of a dynamic cost calculation. As it is very important to reduce the visual complexity of an AG to enable security professionals and system administrators to fully understand the information presented to secure Network systems from intended attackers.

4.2.4 Vertex ranking

As the AG suffers from visual complexity due to the huge number of vertex and arcs are used to construct the graph, which lead to the lack of in-depth understanding of information offered, different ranking techniques using link analysis have been used to deal with these problems. PageRank [Arasu, Arvind, et al., 2002] metrics is the most respected ranking algorithm, considered as a story of success of Google. Other ranking algorithms considered to be less poplar like Hyperlink Induced Topic Search (HITS) [Ding, Chris, et al., 2002],

Hub-Rank, Sym-Rank and Auth-Rank all use Stochastic methods for link analysis.

Recently, researchers suggested ideas to solve some problems in PageRank such as Input/output efficiency matrix's Eigenvalues calculations to be implemented in AG, in [Rungsawang, Arnon, and Bundit Manaskasemsak, 2003] Many examples demonstrated that during the calculations of derived eigenvectors in the directed graphs matrices, it is possible to derive the ranks within the sub – link analysis.

This approach is used in the CostRank algorithm to rank the dynamic cost-centric AG. The main idea of the CostRank algorithm in terms of vertex ranking is to consider a specific state (vertex) as a high rank state if it is linked with many other high rank states.

A fixed probability distribution, conversion matrix is used in graph ranking [Sawilla, R., and X. M. Ou, 2007] calculations, The graph is primarily mapped to a random walker matrix, occasionally called random jump, as each vertex/node is assigned with a value based on the probability of incoming/outgoing links representing the relationships with other vertex. Referring to Perron's theorem, that makes the principal eigenvector unique, positive, and real. Mehta et al. [Mehta, Vaibhav, et al., 2006] gives thorough descriptions of stochastic process models and random walk matrix, which describe the fundamental concepts of the CostRank for vertex ranking.

There are no assurances that the principal eigenvectors will be unique, if the adjacency matrices [Chen, Yen-Yu et al., 2002] [Zonghua et al., 2012] of graphs are not irreducible (the matrix is irreducible if their elements are tightly coupled). SALSA and HITS [Ding, Chris, et al., 2002] are examples of two Web Pages graphs ranking algorithms, which do not use irreducible matrix. In the CostRank algorithm, several principal eigenvectors are used, the results of biggest eigenvalue multiplicity; the principal eigenvalue has been always one of probability evolution methods.

These ranking algorithms compute their output with the use of an iterative power technique; the principal eigenvector provides the ordinary probability for a random walker to be assigned to every vertex. Calculating the fixed distribution employing the power method provides an instinctive form of a random walk.

The calculations of CostRank are based on the stochastic Markov model, as the computation also depends on finding a fixed probability distribution for a Markov chain in which states/websites are represented as vertices or nodes. The CostRank algorithm assigned initial ranking values for each and every vertex/node in the directed graph and creates the initial ranking values matrix. These values will be equivalent to 1/ total number of nodes; it will then create a random walker matrix as each vertex/node assigned with a value based on the probability of incoming/ Outgoing links representing the relationships with other vertex.

In the first iteration, the CostRank algorithm will multiply the initial ranking values matrix by the random walker matrix to get a new ranking matrix, as each vertex will receive a new ranking value.

These processes/iterations will carry on until fixed probability distributions for the Markov chain is held; the ranking matrix will represent the final ranking values for all peaks in the DCCAT.

4.3 THE COSTRANK FRAMEWORK

The CostRank algorithm is developed based on Google PageRank algorithm to demonstrate significance high rank states based on dynamic cost probability that the attackers can use in multi-step attack to exploit a vulnerability in the target critical asset.
As demonstrated in Figure 4-7, the core of the security solution presented in this thesis is CostRank method.

The CostRank framework is presented in this section to map the thesis objectives with different phases of the process of producing important high rank states to reduce the complexity of the classical AG and to present useful information can be used by the security professionals to secure network systems.

1- Activated the Nessus scanners for all the hosts in the network system to reveal the individual vulnerabilities are exist in each host.
2- Combine the Network configuration file with devices rules file to create a single network specification file (Network. xsl) contains the necessary information of network architectures and configurations to be employed in the DVSS calculation.
3- Compute the DVSS score for each vulnerability existing in the network hosts, at this phase objective 1 was met.
4- Put through the cost-centric method to specify a unique severity cost (impact cost-centric) for each host by dividing the loads of the vulnerabilities to three principal degrees of privileges: (i) none; (ii) user and (iii) root, and then sorted out these levels into operational levels to key out and estimate the severity cost of multi-step vulnerabilities.
5- Construct the DCCAG according to the cost-centric values, which represents the states of each attack actions of the vulnerabilities, at this phase objective 2 was met.
6- Run CostRank algorithm using stochastic Markov model calculations taking into account the dynamic cost-centric approach via matrix's Eigenvalue calculations to rank the states of each attack actions of the vulnerabilities using statistical probability of the dynamic vulnerability impact model. This approach enhanced the classical approaches of constructing and analysis of an AG, at this phase objective 3 was met.

As you can see from Figure 4-7, there is a need to develop a new parallel algorithm to implement CostRank for distributed parallel

computers using multiprocessors. To reduce the complexity of the serial CostRank algorithm, that related to the objective four and will be discussing in Chapter-5.

All the phases mentioned above will be used to develop an effective countermeasures solution against network attacks using cost-centric model, this work related to objective five and will be covered in Chapter-6.

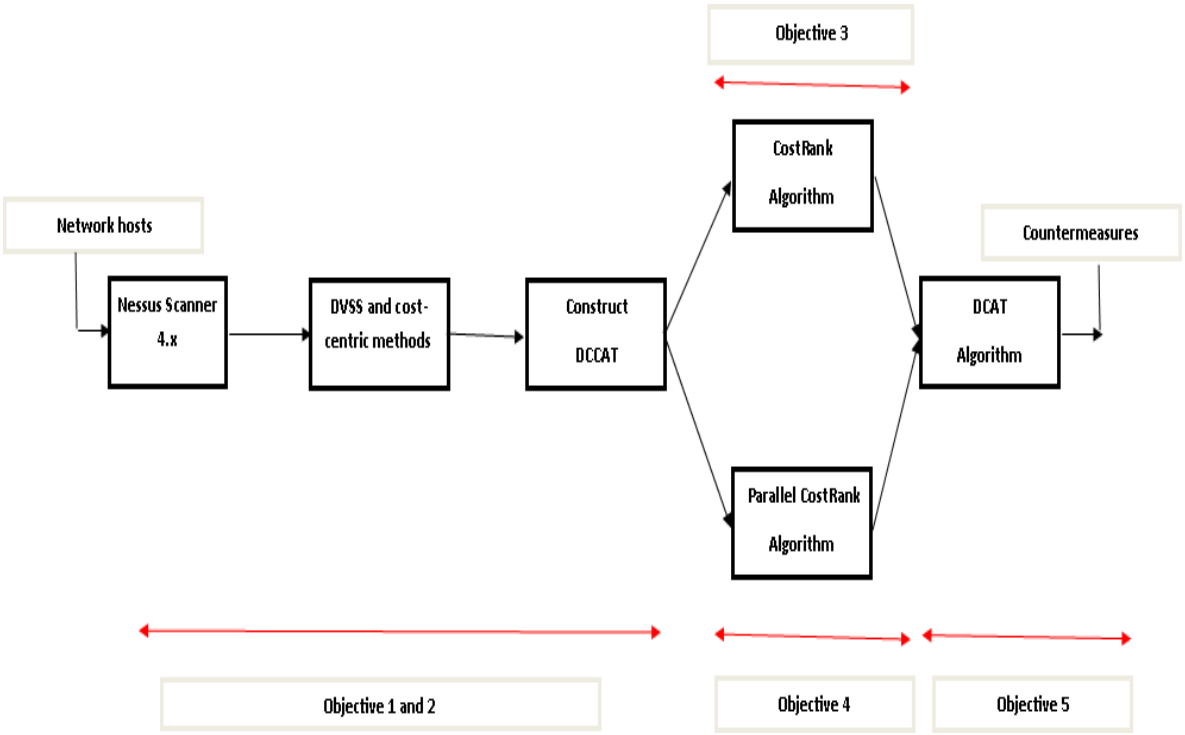


Figure 4-7. The construction of the security solution proposed in this thesis

4.4 QUERY INDEPENDENT LINK BASED RANKING-PAGERANK

A CostRank algorithm is developed based on PageRank using stochastic Markov model calculations taking into account the cost-centric approach via matrix's Eigenvalue scheming, This was used to rank the states of

vulnerabilities using a statistical probability mitigation to a dynamic vulnerability impact model.

The main alterations, between the Costrank and the PageRank approach are presented in Section 5.9.

The following definition of PageRank *(ℜ) is given.* Let u be a web page (a vertex in the graph). Then let $F_w(u)$ represent the groups of forward (outgoing) links from u and let $B_k(u)$ represent the groups of (incoming) links to u. Given o*d (u)* = $|F_w(u)|$ represent the out degree of u.

$\Re(u)$ represents the rank $\Re$ of vertex *u*. The initial rank Ŗ values of the vertices are computed as follows, each vertex is allocated to the initial Ŗ value equivalent to 1/n, n represents a whole number of vertices/pages in the Webpages directed graph.

$$\Re v = \sum_{u \in B_k(u)} \frac{\Re(u)}{od(u)} \qquad for \qquad \forall u \in V$$

At the conclusion of this phase, the initial ranking vector representing the initial ranking values for all the vertices in the Webpages graph.

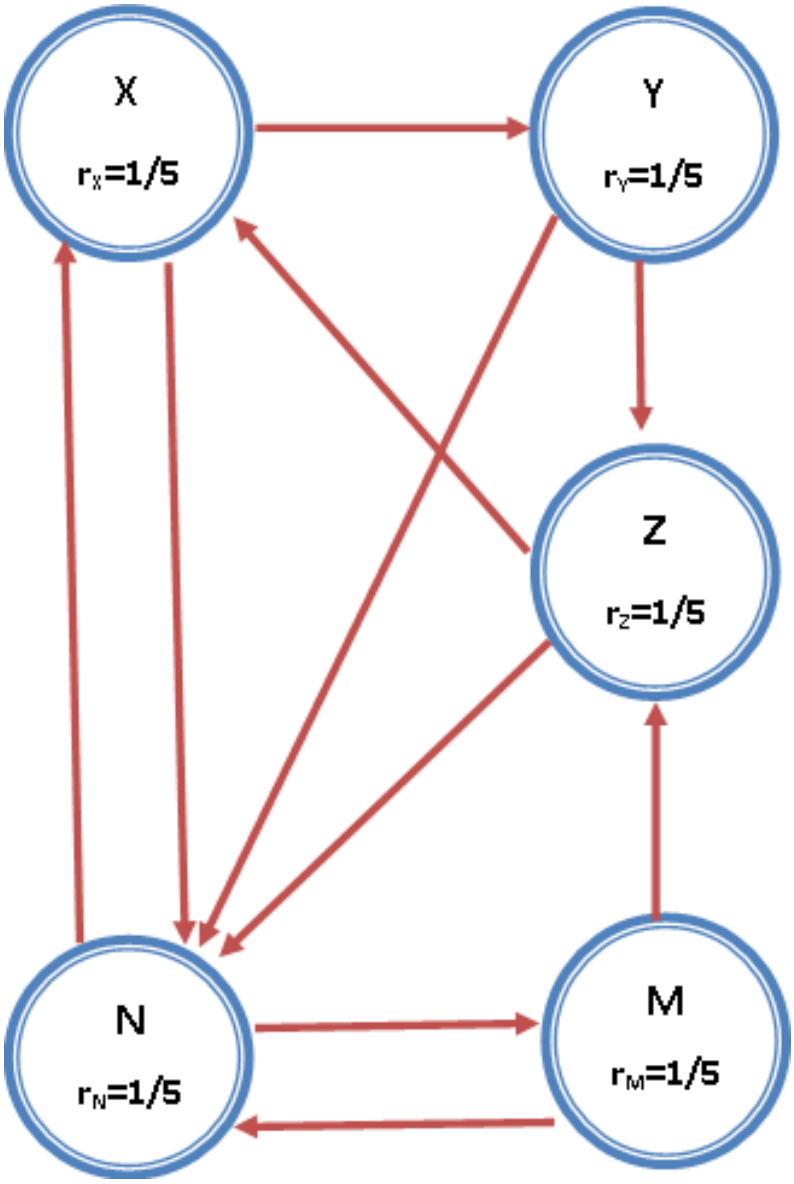


Figure 4-8. A simple web graph with vertex weight

The second phase is the calculation of random walker matrix P_R that stands for the row stochastic matrix related to the Webpages graph G.

In this demo, all arcs assumed to have, in any event, one outgoing connection. In the case exhibited in Figure 4–8, there are outgoing links from page X to page Y, and from page X to page N then the total outgoing links from X is represented by o*d (i)* where i interpret a specific vertex in the graph, in this case od (x) =2. The random walker matrix entries $P_R(xy)$ have the value *1/od (i)*; all the other representations will have the value 0.

P	X	Y	Z	N	M
X	0.00	0.50	0.00	0.50	0.00
Y	0.00	0.00	0.50	0.50	0.00
Z	0.50	0.00	0.00	0.50	0.00
N	0.50	0.00	0.00	0.00	0.50
M	0.00	0.00	0.50	0.50	0.00

Table 4-1. Transition probability of random walker matrix

The third phase is to calculate the new ranking values by using the vector matrices multiplications, by multiplying the initial ranking vector by the random walker matrix as first iteration, the result will be the new ranking matrix $[\Re]_{1n} \times [P_R]_{nn}$. Repeated multiplication of the vector $\Re$ by random walker matrix P_R will give the principal eigenvector of the matrix P_R. The new ranking matrix after iterations represents a stochastic transition matrix of Web pages graph G. The PageRank $\Re^{(t+1)}$ represents the results of vector/matrix multiplications. As explained, as these cancellations represent fixed probability distributions over web page links, the iterations (multiplications) must carry on until the ranking probability reaches the stable states values (fixed) to represent the final vertex ranking in the graph as shown below:

$$\Re^{(t+1)} = \Re^{(t)} * P_R$$

$$\Re^{(0)} \quad * \quad P_R = \quad \Re^{(1)}$$

$$[1/5 \;\; 1/5 \;\; 1/5 \;\; 1/5 \;\; 1/5] * \begin{pmatrix} 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 1/2 & 1/2 & 0 \\ 1/2 & 0 & 0 & 1/2 & 0 \\ 1/2 & 0 & 0 & 0 & 1/2 \\ 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix} = [2/10 \;\; 1/10 \;\; 2/10 \;\; 4/10 \;\; 1/10]$$

The P_R represents the random walker matrix of web browsing over the Webpages graph; the calculation of a fixed probability distribution for a Markov chain of Web pages in the directed graph represents the ranking states. The separate time steps in the chain and the iterations of $\Re$ computation are shown in Figure-4-9. The ranking after 13 iterations, until the algorithm reached fixed probability, is X=0.22, Y=0.17, Z=0.18, N=0.27 and M=0.16. In this case, *N* is the most important web site.

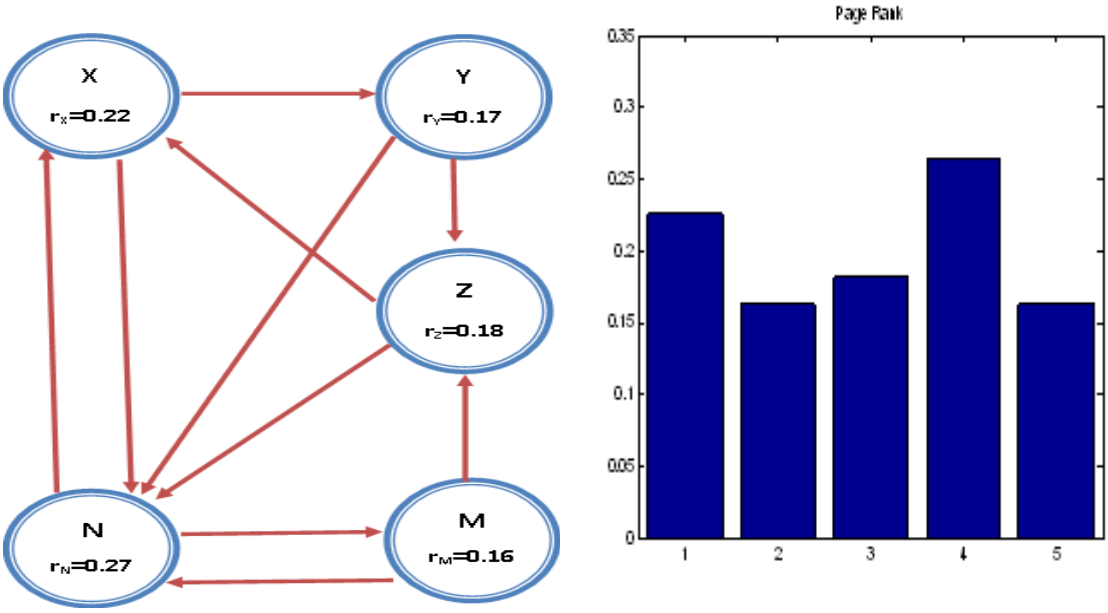


Figure 4-9. A simple web graph with vertex weight after 13 iterations

In the following example shown in Figure 4-10 (a), the damping factor (d), as cited earlier the final probability assigned to a specific webpage represent a specified probability distribution for a Markov chain of all Web pages points to it, to reduce the influence of many pages added their probability values, a value between 0, 1 assigned to damping factor.

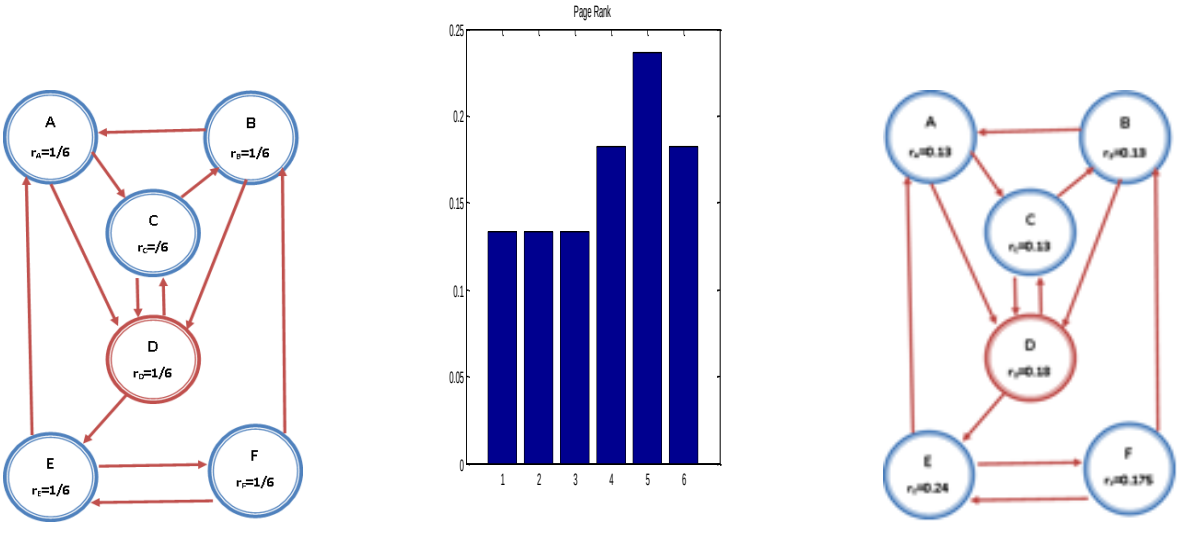
The majority of the articles and practical assignment of google page rank used the value d= 0. 85, but other values could be used according to specific environments. In other, hand the value (1-d) which assigned the value 0.15 to a specific web page when there is no page appointed to it at all.

The pattern used to forecast the probability distribution, modified as follows:

$$\Re^{(t+1)} = (1-d)/d * \sum_{1}^{i} (\Re_i^{(t)} * P_R)$$

The ranking after 13 iterations as shown in Figure 4-10 (b), until the algorithm reached fixed probability, is A=0.135, B=0.135, C=0.135, D=0.18, E=0.24 and F=0.175. In this case, E is the most important web site.

Please note the sum of the all fixed probability of all web pages always equal to one.



(a) (b)

Figure 4-10. A network graph with vertex weight using damping factor after 13 iterations

4.5 METHODOLOGIES FOR DEVELOPING THE DYNAMIC COST-CENTRIC RANKING APPROACH

In real-world environments, the complexity of an AG, is well beyond the human's capabilities to understand. It is important to distinguish applicable parts of the graph to analyse. The ranking model of an AG is suggested to accomplish that. In the CostRank approach, a ranking model of the dynamic states of probabilistic cost-centric AG is proposed similar to the Google PageRank algorithm to demonstrate significantly higher ranked states based on dynamic cost probability that the attackers can use in multi-step attack to exploit a vulnerability in the target critical asset. Using a CostRank algorithm for cost-centric AG, can allow the security professional and system administrator to focus on specific vulnerabilities, thus is allowing them to deploy a policy of implementing security countermeasures to protect the network system in a cost effective manner.

In the dynamic CostRank approach, instead of specifying a state by network attributes, a dynamic cost- centric metric is proposed; each state in the graph represents a dynamic cost attributes of all vulnerabilities in a single host as explained in chapter-3. The resultant attack graph is called a Dynamic Cost Centric Attack Graph (DCCAG), which take into considerations the network architectures and configurations, along with the interactions between the individual vulnerabilities, Along with a new novel CostRank algorithm that reduced the complexity and provides security professionals with an effective tool to enhance the security.

As shown in Figure 4-13, the principal sign for the CostRank algorithm to be revealed, a state is important if other important states are linked to it. However, not all states are equal in importance (cost); the link from different states holds a different cost value. The initial value vector is assigned the cost derived from a dynamic cost framework for each state in the graph, which describes the initial

CostRank value for all states V_i. Then the excursive formula is iterated until two consecutively iterated CostRank vectors are similar enough [Gürdag, Adnan Burak, 2002]. In the CostRank algorithm, the input is not directly taken from link files (as explained in more detail in chapter (5)); instead, the out-link file is converted into a binary graph file structure as demonstrated in Figure 5-1. The total number of floating point numbers read from graph file is |*2*Vertexes (V_x)*| at the start of an execution. While the size of the data read from an attack graph file during each iteration is $|2*|\acute{E}|+2*|(3+s)*V_x\|$ integers, where s represents the number of out links for each state.

In the developed algorithm shown in Figure 4-13, $\Re^{(t)}$ represents the CosrRank of the state x and n is the number of links point to state x and N is the total number of states in a network representation graph.

$$\Re^{(t+1)}(x) = ((1-d) - \frac{n}{\sum od_i} * \frac{1}{N}) + \sum_{1}^{i} (\Re_i^{(t)} * P_R) \quad \text{for Initial attack state}$$

$$\Re^{(t+1)}(x) = (d - \frac{n}{\sum od_i} * \frac{1}{N}) + \sum_{1}^{i} (\Re_i^{(t)} * P_R) \quad \text{otherwise.}$$

If the attacker in the Initial stage of multi-step attack, in this case the attacker, will commence with a probability of (1-d), otherwise if the attack in the middle of the terminal stage of attack will continue with probability d.

In this model, the attributes for any vertex X includes:

1- The right of entry (access) or gaining privilege of the attacker on host X: Classified by privilege levels, such as:
   - System administrator or the Root privilege (2).
   - Users privilege (1).
   - None privilege (0). (The attacker has no privilege on the node.)
2- Security dynamic metrics, cost for each host: represent the CVSS score for each host.
3- Exploit style: The location of an attacker during an exploit (LAN, remoteness network, WANS).

To illustrate the proposed approach in detail, consider a prototype network as shown in Figure 4-11 taken from [Mehta, Vaibhav, et al., 2006] and used by [Kijsanayothin, Phongphun, 2010] to compare the results, where readily available are two service hosts, IP1, IP2, and an attacker's workstation; the attacker connects to each of the servers via a central router.

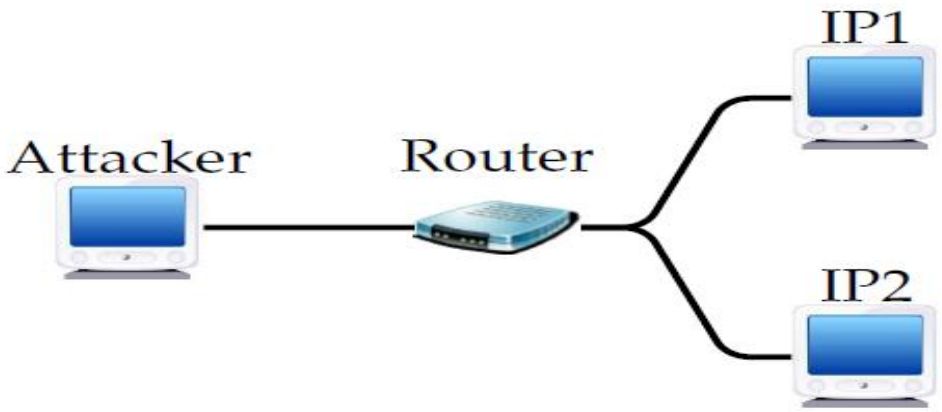


Figure 4-11. A prototype network taken from [Mehta, Vaibhav, et al., 2006]

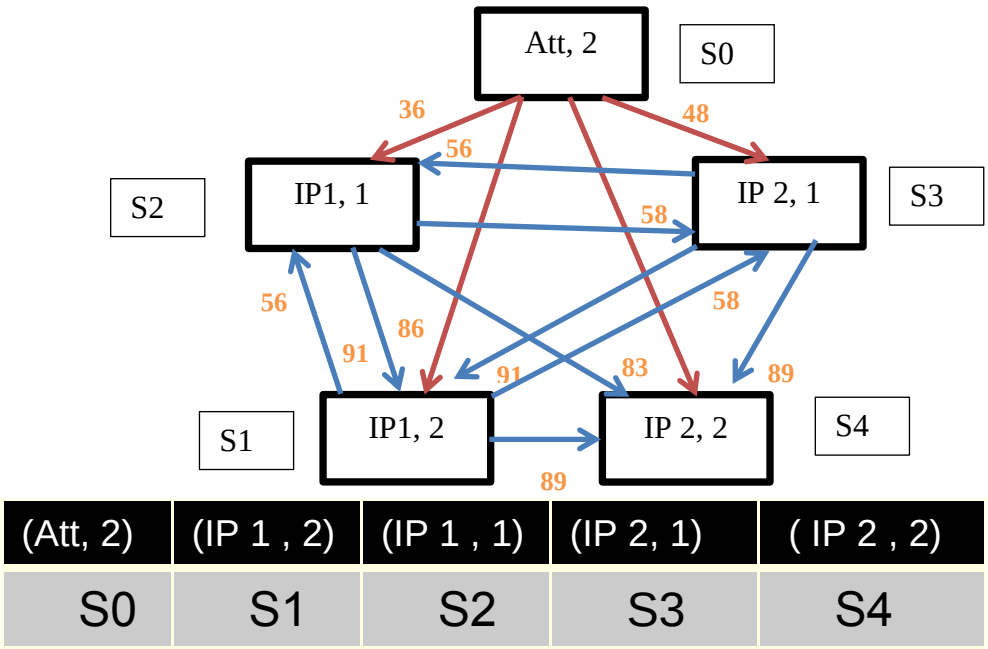


Figure 4-12. Cost-centric attack graph with arc weight

/* TPM = Transitions Probability Matrix.

/* G=The graph.

/* d= Damping factor.

/* Cost=CostRank (G , TPM, d)

[r, q]= size (G);

/* od= outlinks; id= inlinks

od=sum (TPM, 1);

Id= sum (TPM, 2);

K=0; s=0;

for i:= 1 to r do

If s=0 /* Initial attack state

$$\Re^{(t)}(x) = ((1-d) - \frac{n}{\sum od_i} * \frac{1}{N}) + \sum_{1}^{i} (\Re_i^{(t+1)} * TPM_i)$$

else

$$\Re^{(t)}(x) = (d - \frac{n}{\sum od_i} * \frac{1}{N}) + \sum_{1}^{i} (\Re_i^{(t+1)} * TPM_i)$$

k++

If (k <= n) & (x (q (k)) >= .005)

break;

end

return cost = $\Re^{(t)}(x)$

end

Figure 4-13 The CostRank algorithm

	S0	S1	S2	S3	S4
S0	0	86	36	48	83
S1	0	0	56	58	89
S2	0	91	0	58	83
S3	0	91	56	0	89
S4	0	0	0	0	0

Table 4-2. Cost matrix representations

As shown in Table 4.2, the calculated cost values derived from the DVSS and operational level cost-centric framework (as explained in Chapter-3 of this thesis) recorded for each state in the graph as cost matrix.

In the normalised cost matrix, the following formula is used:

$$\Re^{(t+1)}(x) = \frac{C^{(t)}(x)}{\sum_{i=1}^{n} \Re^{(t)}(i)}$$ where n is the number of links point to state x

For example $\Re^{(t+1)}(s0 \to S1) = \frac{C^{(t)}(s0 \to S1)}{\sum_{i=1}^{n} \Re^{(t)}(i)} = \frac{86}{86+36+48+83} = 0.339921$

	S0	S1	S2	S3	S4
S0	0	0.339921	0.142292	0.203390	0.351695
S1	0	0	0.275862	0.285714	0.438424
S2	0	0.392241	0	0.250000	0.351695
S3	0	0.385593	0.237288	0	0.377119
S4	0	0	0	0	1

Table 4-3. The normalised cost matrix

Let $\Re^{(t)}(x)$ be the probability of intrusion of attack state x at time t and d be a damping factor representing the probability of an attacker continuing to penetrate the current path of the attack. Thus, in the context of network:

$$\Re^{(t+1)}(x) = ((1-d) - \frac{n}{\sum od_i} * \frac{1}{N}) + \sum_{1}^{i} (\Re_i^{(t)} * P_R) \text{ for Initial attack state}$$

$$\Re^{(t+1)}(x) = (d - \frac{n}{\sum od_i} * \frac{1}{N}) + \sum_{1}^{i} (\Re_i^{(t)} * P_R) \text{ otherwise}$$

where n is the number of links point to state x and N is the total number of states in a network representation graph.

Since this is a recursive formula, the iterative implementation will make the calculation faster, and will require several iterations before stabilising to an acceptable solution. This can be solved in an iterative fashion using the algorithm shown in Figure 4-13.

	Mehta et al	Kijsanayothin	CostRank
S0	0.1500	0.1500	0.1500
S1	0.1450	0.1287	0.210806
S2	0.1020	0.1658	0.123615
S3	0.2090	0.2548	0.139394
S4	0.3940	0.3007	0.475065

Table 4-4. Comparing CostRank Ranking results with Mehta et al. and Kijsanayothin's approach

Table 4-4 shows the ranking result obtained when the experimentation is done, that is s4, s1, s0, s3, s2 (i.e. CostRank approach).

Otherwise, s4, s3, s0, s1, s2 is obtained when Mehta et al.'s approach is applied and s4, s3, s2, s0, s1 when Kijsanayothin's approach is used.

4.6 INITIAL RESULTS AND DISCUSSION

The ranking produced is in the order of likelihood of intrusion based on exploitability. For example, all results suggest that s4 has the highest (relative) likelihood of being attacked (i.e. highest exploitability and most vulnerable), to further compare the ranking results, if s0 is omitted in all ranking lists, both ranking orders generally agree except for the conflicting case of ranking orders between s1, s2 and s3.

1-{s1, s2}: Consider attackers from the initial state. As shown in Figure 4.12 to reach state s1 (e.g., from s0, s2 or s3) requires exploiting cost 89 71 85, whereas to reach state s2 (e.g. from s0 or s3) requires exploiting cost 79 85. However, according to simple calculations, s1 is more vulnerable than s2; in addition to reaching s1 means, the attacker has the ability to compromise a host

and reach the root of IP1; this is not the case for s2. Therefore, s1 should rank higher than s2.

2-{s2, s3}: starting from an initial state s0, to reach s4 through s2 requires three applications of the costs of exploitability value 79 49 70 , whereas to reach s4 through s3 only requires two applications of the costs of exploitability value 49 70. Therefore, s3 should rank higher than s2.

3-{s1, s3}: Consider attackers from the initial state. To reach state s1 (e.g., from s0, s2 or s3) requires exploiting cost 89 71 85, whereas to reach state s3 (e.g., from s0 or s2 or s1) requires exploiting cost 49 49 66. Therefore, s1 should rank higher than s3 because s1 is more vulnerable than s3.

4- {s3, s4}. If there is only one-manner to reach s4 by an attack path from s0 to s3 to s4 then s3 ranks higher than s4. However, s4 can be reached by another attack path from s0 to s2 to s4 and attack path s0 to s3 to s4, thus, s4 ranks higher than s3.

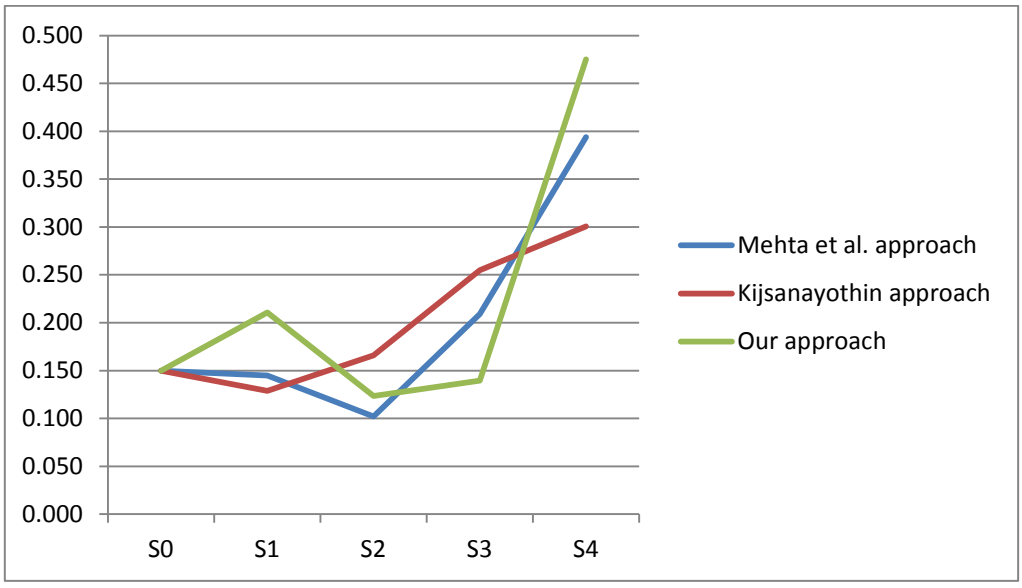


Figure 4-14. Initial cost-centric ranking results

As shown in Figure 4-14, as a direct result of employing the dynamic approach in the process of estimating the cost value of exploiting each state. In the graph, the ranking results of DCCAG are more effective, represent the

substantial impact of vulnerabilities, evidenced that the CostRank model is more effective in term of detecting the actual risk, and can be utilised to enforce an efficient method to implement the countermeasures and mitigations.
The resolutions above certified that the CostRank algorithm and approaches produce accurate solutions in terms of ranking the states during any attack compared to the other two methodologies. The system administrator should give first priority to s4 (IP 2, 2) {compromised root of IP2} as it possesses the highest (relative) likelihood of being attacked. It has the highest exploitability and is the most vulnerable. Resolutions should be immediately implemented, then in the second place s2 (IP 1, 2) {compromised the root of IP1} then s3 (IP 2, 1) then s2 (IP 1, 1) as explained.

4.7 INVESTIGATIONS

In the next case, the algorithm has been carried out on a relatively small prototype network as shown in Figure 4-15. This example represents LAN exploit style; to commence with the attacker attached to host- A and needs to exploit a vulnerability at host-E or Host-F, both of which are protected by a firewall.

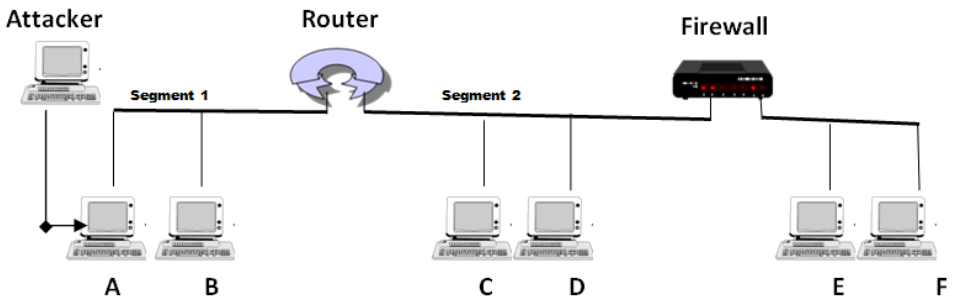


Figure 4-15. An example network, modified from [Franqueira et al., 2009]

The firewall rules just permit communications of host-C or host-D to host-E. Router (Ro) connects segments 1 and 2. Additionally, all hosts have a CVE-

2003-0818: a vulnerability in the single service called "*Microsoft Windows ASN.1 Library Integer*."

Gaining system administrator or root privilege results if the attacker manages to exploit the vulnerability, buffer overflow state result of sending large packet, which is achieved by running arbitrary code in it to get root privileges, potentially having a comprehensive impact/damage on its availability (A), confidentiality (C), integrity (I). A full-scale scenario CCAG is presented in Figure 4-16.

Table 4-5 shows the ranking result obtained when the CostRank algorithm castoff in the experimental approach using the normalised cost matrix.
Using the CostRank algorithm, in the first iteration T1 the ranking values are calculated by matrix multiplications of the initial ranking values vector by the random walk matrix. This process must carry on (normally for 13 iterations) until the ranking probability reaches the stable status values (fixed). The excursive formula is iterated until two consecutively iterated CostRank vectors are similar enough, as you can see in Table 4-5, in this case the two consecutively iterated CostRank vectors are similar enough at T13, where the state values reach the fixed value, the CostRank values presented for 24 iterations, please refer to Appendix -F for all iterations.

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12
T1	0.0375	0.0442	0.0438	0.0352	0.0463	0.0608	0.0768	0.0584	0.0719	0.1001	0.1337	0.1413
T2	0.0294	0.0215	0.0252	0.0200	0.0488	0.0635	0.0607	0.0583	0.0546	0.1001	0.1740	0.2456
T3	0.0478	0.0169	0.0123	0.0115	0.0488	0.0635	0.0661	0.0461	0.0545	0.1001	0.1740	0.3197
T4	0.0188	0.0275	0.0096	0.0056	0.0488	0.0635	0.0661	0.0502	0.0431	0.1001	0.1740	0.3197
T5	0.0434	0.0513	0.0501	0.0386	0.0523	0.0701	0.0890	0.0673	0.0825	0.1150	0.1559	0.1642
T6	0.0435	0.0514	0.0503	0.039	0.0526	0.0703	0.0892	0.0675	0.0828	0.1154	0.1561	0.1645
T7	0.0436	0.0515	0.0505	0.0394	0.0529	0.0705	0.0894	0.0677	0.0831	0.1158	0.1563	0.1648
T8	0.0437	0.0516	0.0507	0.0398	0.0532	0.0707	0.0896	0.0679	0.0834	0.1162	0.1565	0.1651
T9	0.0438	0.0517	0.0509	0.0402	0.0535	0.0709	0.0898	0.0681	0.0837	0.1166	0.1567	0.1654
t10	0.0439	0.0518	0.0511	0.0406	0.0538	0.0711	0.09	0.0683	0.084	0.117	0.1569	0.1657
t11	0.039	0.0519	0.0513	0.041	0.0541	0.0713	0.0902	0.0685	0.0843	0.1174	0.1571	0.166
t12	0.0440	0.051	0.0514	0.0413	0.0542	0.0714	0.0903	0.0686	0.0844	0.1176	0.1572	0.1662
t13	0.0441	0.052	0.0515	0.0414	0.0544	0.0715	0.0904	0.0687	0.0846	0.1178	0.1573	0.1663

Table 4-5. The ranking result obtained when CostRank algorithm castoff in the experimental approach using the normalised cost matrix

The final ranking results driven from the cost matrix are presented at a lower place:

s12, s11, s10, s7, s9, s6, s8, s5, s3, s2, s1, s4.

S(0)	S(1)	S(2)	S(3)	S(4)	S(5)	S(6)	S(7)	S(8)	S(9)	S(10)	S(11)	S(12)
0.076	0.044	0.052	0.051	0.041	0.054	0.071	0.0904	0.0687	0.0846	0.1178	0.1573	0.1663

Table 4-6. Experimental network ranking results

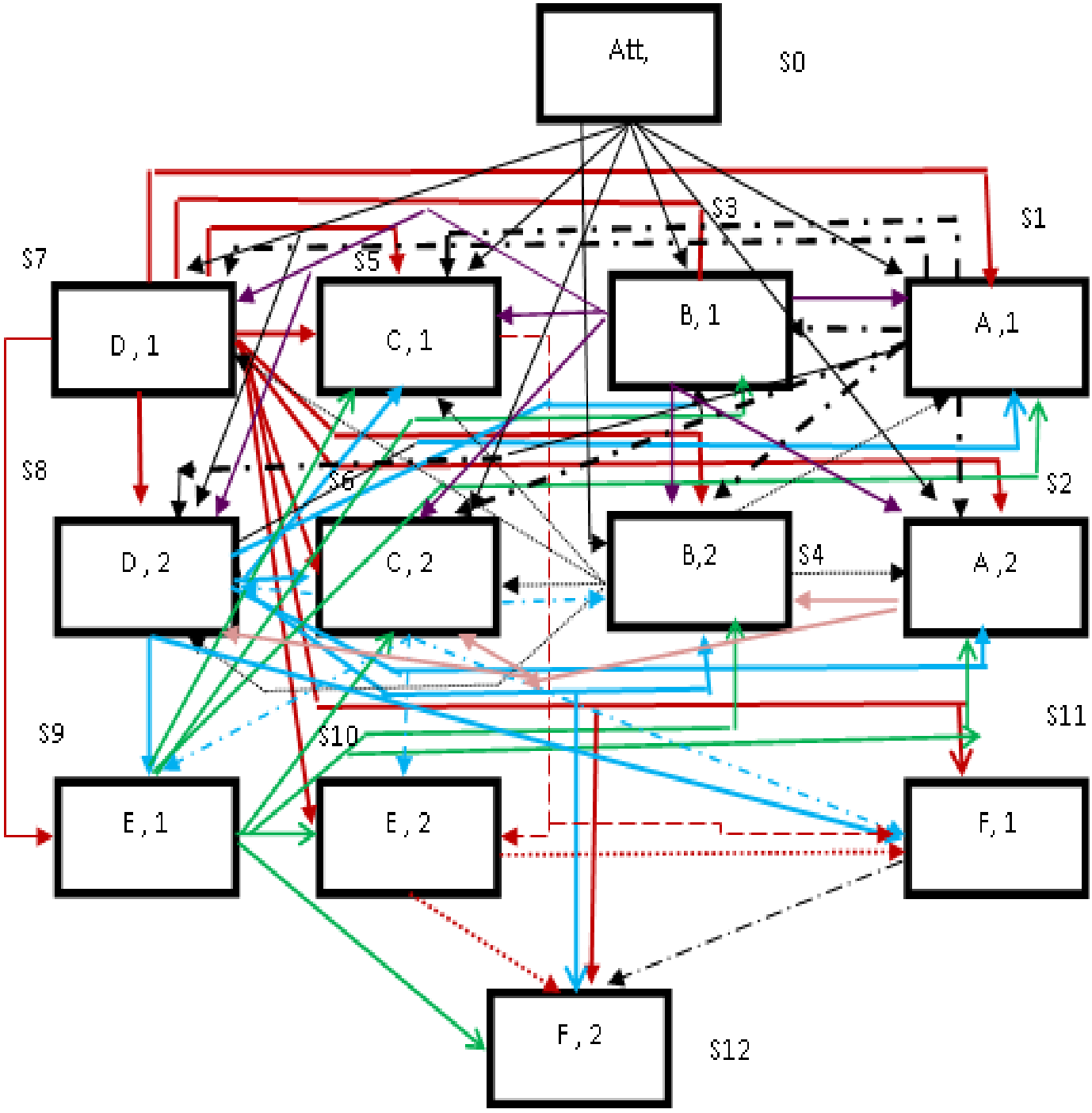


Figure 4-16. A full-scale scenario of a prototype network

4.8 RESULTS JUSTIFICATION

1.{s12, s11} s12 can be reached by another attack path to s4 thus, s12 ranks higher than s11.

2.{s11, s10} total cost to reach s11: 87 85 87 88 72 69 69 = 557, but the total cost to reach s10 from initial state = 482. Therefore, s11 should rank higher than s10 (same segment).

3.{s9, s7} the total cost to reach s9 = 374 and to s7 = 528 from the initial state. Therefore, s7 should rank higher than s9.

4.{s9, s8} because s9 (f1) is in the same segment while s7 (c1) is in the other segment, the firewall rules only permit communications of host-C or host-D to host-E. Therefore, s9 should rank higher than s8.

5.{s6, s7, s8} total cost to reach s6 = 434, and the total cost to reach s7 from initial state = 528, the total cost to reach s8 from initial state = 407. Therefore s7 (cm2), should rank higher than s6, s8 and s6 rank higher than s8 (same segment).

6.{s6, s5} total cost to reach s6 = 434, while the total cost to reach s5 from initial state=334. Therefore, s6 should rank higher than s5, especially s6 = (d, 2) compromised the root (same segment).

7.{s5, s4} total cost to reach s4 = 209 and to s5 = 334 from the initial state. Therefore, s5 should rank higher than s4.

8.{s3, s2} from initial state. Therefore, s3 should rank higher than s2.

9.{s1, s2} total cost to reach s2 = 257 and to s1 = 222 from the initial state. Therefore, s2 should rank higher than s1.

10.{s1, s2} total cost $\sum cost$ to reach s2= 257 and to s1 = 222 from the initial state. Therefore, s2 should rank higher than s1.

4.9 CONCLUSION

The Classical AG, provide the security professionals and system administrators with many tasks to fix the existing security problems and require them to carry through the AG model again, if something sounded awry. As the classical AG represents a range of individual vulnerabilities with static natures because, the interactions between the vulnerabilities and mostly the network topology and device configurations are not counted.

In this thesis, formal techniques are built up for the dynamic cost-centric AG, the courses of an AG are representing a chain of the states of dynamic exposures.

In the dynamic cost-centric approach, instead of specifying a state by network attributes, a dynamic cost- centric metrics are proposed; each province in the graph represents a dynamic cost attributes of all vulnerabilities in a single host as explained in chapter-3 satisfying objective 1 and 2 of this thesis. The resultant attack graph is called a Dynamic Cost Centric Attack Graph (DCCAG), which takes into considerations the network architectures and configurations, along with the interactions between the individual vulnerabilities,

A new novel CostRank algorithm, to examine objective 3 of this thesis, which cut the complexity and offers security professionals with an efficient tool to heighten the security.

A scalable CostRank algorithm for ranking the AG uses a novel method to represent the states by implementing a dynamic cost-centric driven from statistical probability of dynamic impact scoring for the exposures.

The ranks of the states represent the importance of the states in the DCCAT. The resulting ranking represents a metric that can be used by security professionals and organisation administrators to establish different security decisions in order to improve the network security based on monetary value/benefits as demonstrated in the business case subject to examine objective 4, 5 and 6 in this thesis (please refer to Chapter-6).

Ranking an AG signifies a useful approach to solve the problem of an AG visual complexity by providing an insight into the critical area of vulnerabilities to perform the analysis and to discover the effective method to implement the mitigation required.

A new methodology is developed to represent attack graphs with cost metrics and model the problem's parameters into a mathematical framework. Instead of specifying a state by network attributes, a dynamic cost-centric model is proposed, where each state in the attack graph represented a severe cost of all vulnerabilities existing in the node (host).

The CostRank algorithm is developed based on Google PageRank algorithm to demonstrate significance high rank states based on dynamic cost probability that the attackers can use in multi-step attack to exploit a vulnerability in the target critical asset. The principal suggestion for the CostRank algorithm to be considered a state is important if other important states are linked to it. However, not all states are equal in importance (dynamic cost), the link from different states holds a different cost value. The initial value vector is assigned the cost derived from a dynamic cost framework for each state in the graph, that describe the initial CostRank value for all states Vi, Then the excursive formula using stochastic Markov model calculations is iterated until two consecutively iterated CostRank vectors are similar enough. In the CostRank algorithm, the input is not directly taken from link files; instead, the out-link file is converted into a binary graph file structure. In the CostRank algorithm the information about pre-condition probability, not essentially to be known for all states as the value can be alleged to perform the ranking, however knowing the probability can produce more accurate results.

The main differences between PageRank and CostRank are:

1- PageRank uses probabilities derived from the initial number of pages and Markov model calculations of consecutively iterated, in CostRank The initial value vector is assigned the cost derived from a dynamic cost framework for each state in the graph, that describe the initial CostRank value for all states Vi. In terms of complexity the PageRank only considers the number of pages

pointing to another page using fixed probability distribution, conversion matrix in the process of rank calculations. The web graph is primarily mapped to a random walker matrix, as each vertex/node is assigned with a value based on the probability of incoming/outgoing links representing the relationships with other vertex, while in CostRank the consideration should be given to the privileges, dynamic cost and exploit style.

2- In PageRank, the damping factor (d) is assigned to a specific when there is a page(s) pointed to it while the value (1-d) which assigned to a specific web page when there is no page appointed to it at all. In CostRank, If the attacker in the Initial attack stage of multi-step attack in this case the attacker will start with a probability of (1-d), otherwise if the attack in the middle of the final stage of attack will continue with probability d.

3- The CostRank algorithm exercises an iterative mathematical process to calculate the maximal eigenvector of a practiced matrix with hosts' dynamic cost values derived from designated file structures.

A framework is carried out on a test network, using the Nessus scanner to detect known vulnerabilities, implement these results and to build and represent the dynamic cost centric attack graph using ranking algorithms (in a standardised fashion to Mehta et al. 2006 and Kijsanayothin, 2010). However, instead of using vulnerabilities for each host, a dynamic CostRank Markov model based on the states has developed. As a direct consequence of a framework implementation using the dynamic advance in estimating the Costs value of exploiting each state in the graph. The ranking results of DCCAG are more effective and represent the substantial impact of vulnerabilities. That has evidenced the CostRank model is more effective in term of detecting the actual risk and can be utilised to enforce an efficient method to implement the countermeasures and mitigations. The results, evidenced that the CostRank algorithm and approaches produce accurate solutions in terms of ordering the states during any attack compared to the other two methodologies, as well reducing the complexity in the attack graph and reducing the problem of visibility

In the following chapter, a new parallel algorithm is applied to implement CostRank for distributed parallel computers using multiprocessors, to reduce the complexity $\Theta(n^2)$ of the serial CostRank algorithm. As the number of hosts in network are increased, the serial CostRank algorithm is need to read the source CostRank values of different states and store it in a buffer, then read the parameters in the binary graph file header, and multiply the ranking vector of the random walker matrix.

To make CostRank calculations are more effective in term of a cost/benefit indication. An analysis of the performance and scalability of the parallel CostRank algorithm are introduced; the empirical approach to evaluate the CostRank algorithm's performance indicates a major decrease in communication overhead and in runtime.

CHAPTER 5. A PARALLEL ALGORITHM TO CALCULATE THE COSTRANK OF A NETWORK

5.1 INTRODUCTION

The CostRank algorithm is used to rank the states of the DCCAG model based on cost-centric approach for network security. It computes the maximal eigenvector of a random walker Markov chain matrix (transition matrix) with the states' cost value derived from designated file structures. The principal indicator for the CostRank algorithm to consider a state is important if other important states are linked to it. However, not all states are equal in importance; the links from different states hold different cost values. For n states v = 1, 2. . . n. The corresponding CostRank is set to C=C_1, C_2, C_3..C_n. The mathematical formulation for the recursively defined CostRank is presented in Chapter 4. The initial value vector is calculated by following this formula: 1/V; where V is the number of states present that describe the initial CostRank value for all states Vi, and the dynamic costs used to construct the random walk matrix. Then the formula is iterated until two consecutively iterated CostRank vectors are similar enough. The CostRank algorithm does not directly take input from link files, but rather involves the conversion of the out-links file into a binary graph file structure, M, as illustrated in Figure 5-1. The total number of floating point numbers read from graph file is |2*Vertices (V_x) | at the start of an execution. While the size of the data read from an attack graph file during each iteration is | $2*|\acute{E}|+2*|(3+s)*V_x\|$ integers, where s represents the number of out links for each state. Towards the end of the execution, entire |$\aleph$| numbers representing the consequential CostRank vectors have written into disks is:

$$\aleph = 2*|V_x| + (\Re * |\ 2|\acute{E}| + 2*|(3+s)*V_x)|\ |$$

where $\Re$ is the number of iterations. There is a linear association between the amount of links and sum of states in a graph file, and since the number of

iterations is nearly constant, the total time spent on disk I/O during an execution is O(n) where n = |ℜ|.

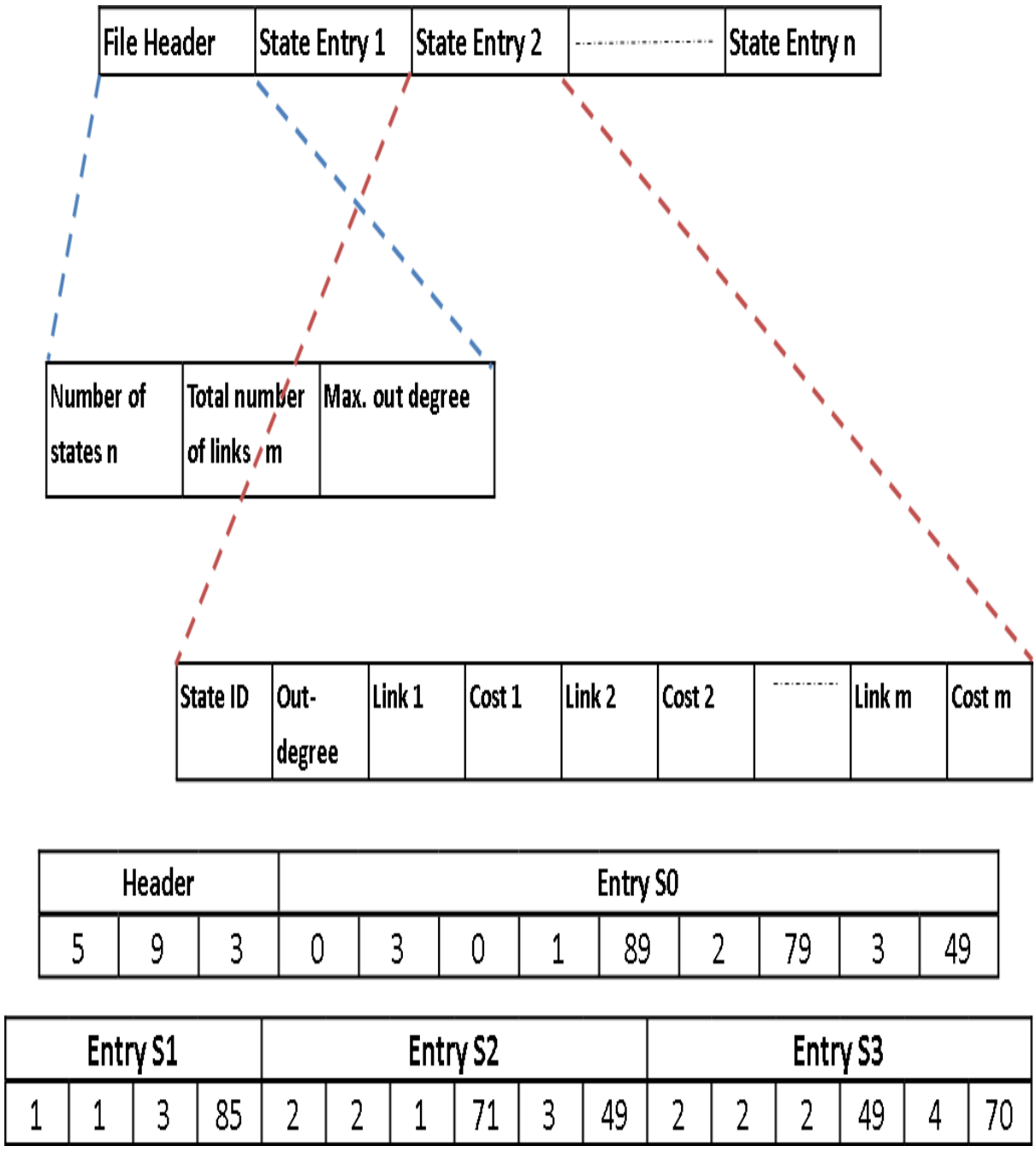


Figure 5-1. The construction of a sample graph file

Figure 5-1 illustrates the construction of an attack graph file. This file is structured with a file header and states input information about the hosts in the DCCAG. The file header has three components: entire number of states, entire number of links and the upper limit out degree in the attack graph. A state entry record saves the essential structural data for a state such as state ID, the out-degree, and then forward links (Link 1, Link 2, and Link v) along with the corresponding cost for each link (C_1, C_2, and C_v) respectively. It represents an array of n state IDs and m of links. All of the numerical data are represented in a 4-byte integer buffer. The total volume of attack graph file is = 4 * (2v+2m+2o), where v represents the total number of states, m represents the entire number of links and o represents the maximum out degree. An example of representing states in a graph file is shown in Figure 5-1. As for memory usage, a buffer is used to hold the attack graph file, size of 4 * maximum Out-degree. The setup for the CostRank implementation is done by creating two arrays of floating point values representing the rank directions calculations, as in Figure 5-2, called the $CostRank_{src}$ and the $CostRank_{dest}$. The source and destination vector have V items, where V represents the full amount of states in the attack graph file.

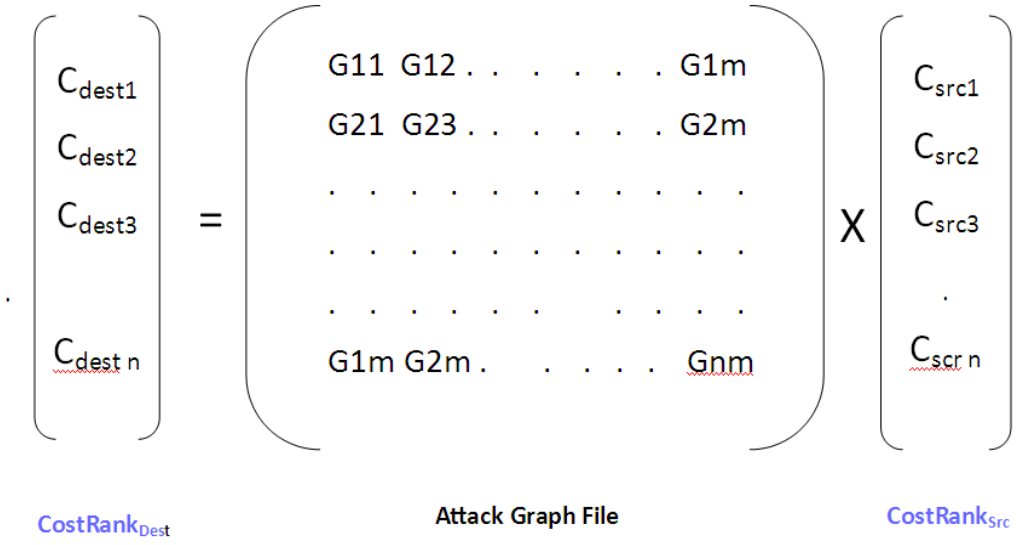


Figure 5-2. The CostRank calculations

Used for each iteration phase, the $CostRank_{src}$ represents the rank value of iteration t and the $CostRank_{dest}$ is referred to as the rank value of iteration t + 1. The sequential algorithm of the CostRank calculation is shown in Figure **5- 4**.

5.2 RESEARCH METHODS

An analogous CostRank parallel algorithm is developed to implement CostRank for disseminating parallel computers with multi processors. To reduce the complexity $(\Theta(n^2))$ of the serial CostRank algorithms as the calculations represent matrix multiplication, as the number of hosts in the networks increased, the serial CostRank algorithm complexity is increased polynomially, as it required matrix multiply. The input n= $(\Theta(n^3))$ to read the source CostRank values of different states and store it in a buffer, then read parameters in the binary graph file header, which has three components: entire number of states, entire number of links and the upper limit out degree in the attack graph. A state entry record saves the essential structural data for a state such as state ID, the out-degree, and then forward links (Link 1, Link 2, and Link v) along with corresponding cost for each link (C_1, C_2, and C_v) respectively. Parallel in-memory Piccolo [Power et al., 2010] algorithm is selected to extensively reduce access share state stored in memory and iterations, which include table partitioning (local access), synchronization of distributed table, checkpoint/restore and load balance and task scheduling. In the same manner, larger scale networks are secured that require quick and reliable computing to calculate the ranking of enormous graphs with thousands of vertices (states) and millions or arcs (links). A parallel CostRank is used, the computational architecture on a cluster of PCs networked via 100 MB/s Ethernet LAN, using a cluster of 32 computers built with Pentium Core2Duo 2.54 GHz CPU, 2GB RAM, 250GB Hard Disk interconnected with 100 MB/s Ethernet LAN, running the Linux operating system. Thirty PCs was used for partitioning calculations and two masters, for synchronisations to assess the complexity and scalability of the algorithms.

In this thesis, the valuation of the parallel CostRank algorithm is based on the performance and scalability. The performance metrics are based on (i) The Total time expended in the CostRank iterations. (ii) Speed up time by running Parallel CostRank over the serial CostRank algorithm. (iii) Efficiencies time stability of CostRank algorithm, measured according to a different problem size. In order to evaluate the scalability, the Isoefficiency Metric is used to perceive the constancy of the efficiency when the problem size and the number of processors increased.

In particular, a partitioning of input data, graph files and ranking vectors with an appropriate load balancing technique has reduced the runtime and hence scalability of large scale parallel CostRank calculations. This thesis presents an application case study of parallel CostRank Calculations using one-dimensional sparse matrix partitioning on a modified research page at Stanford University. It describes the link structure of the stanford.edu-domain from a September 2002 collection and contains 281903 pages with about 2.3 million links, outcomes in a major reduction in communication overhead and in runtime. The performance of the parallel CostRank system, presented to know how have the parallel algorithm done well and right, by measuring the utilisation of the CPU (processor none idle time), the efficiency, the decrease of runtime and speedup metrics. An analytical discussion is provided for the performance of similar algorithms in terms of Execution times and Input/output (I/O), memory usage and synchronization and commendations rate.

The speedup is the proportion of time in use to process CostRank using a single processor (serial CostRank) over the time in use to process CostRank using n processor on Parallel computing process (parallel CostRank). Let suppose that T_s represents the serial time portion of CostRank. The speedup of any parallel algorithm using any number of processors is limited to $1/T_s$, this known as Amdahl's law [Amdahl, 1967], for this reason, the speedup of any parallel algorithm will not continue to rise as the number of processors increased, the speedup be likely to be saturate at a specific number of processors.

The efficiency of the parallel CostRank algorithm calculated using the ratio of speedup to the total number of processors, which measures the portion of time that the processors are busy in processing part of the CostRank calculations.
The scalability of a system measures the change in the performance as the other features of the system is changed such as the problem size, the network bandwidth, the number of processors, and the processor specifications.
The scalability of the algorithm measured according to the its ability to boost the speed up according to the number of processors used, in other words the algorithm's ability to effective granularity on the performance of the processors by maintaining the constant efficiency level while increasing the problem size and the number of processors.
The main goal of the parallel CostRank algorithm empirical approach is to evaluate the scalability and the performance of the parallel CostRank algorithm, the serial CostRank algorithm used as main orientation for evaluation.

5.3 PARALLEL COST RANK ALGORITHM

A parallel algorithm is developed to implement CostRank for distributed memory parallel computers using multi-processors.
Bulk-synchronous parallel model (BSP) projected in 1990 by [L. G. Valiant, 1990.] to solve the weaknesses in the PRAM model is used in the parallel CostRank approach. The model used a fixed number of n processors and memory nodes connected by a computer network, this model used the super step idea of encompassing the synchronisation and communication process, also used public variables or transient message in processor communication.
Piccolo parallel in-memory algorithm is selected [Power et al., 2010] (Piccolo is a programming model which is represented as a library of existing programming languages such as C++, they are used to write in parallel in memory allowing computations from different PCs for parallel implementations). To extensively reduce access share state stored in memory and iterations, which include table partitioning (local access), synchronization of distributed table, checkpoint/restore and load balance/ and task scheduling.

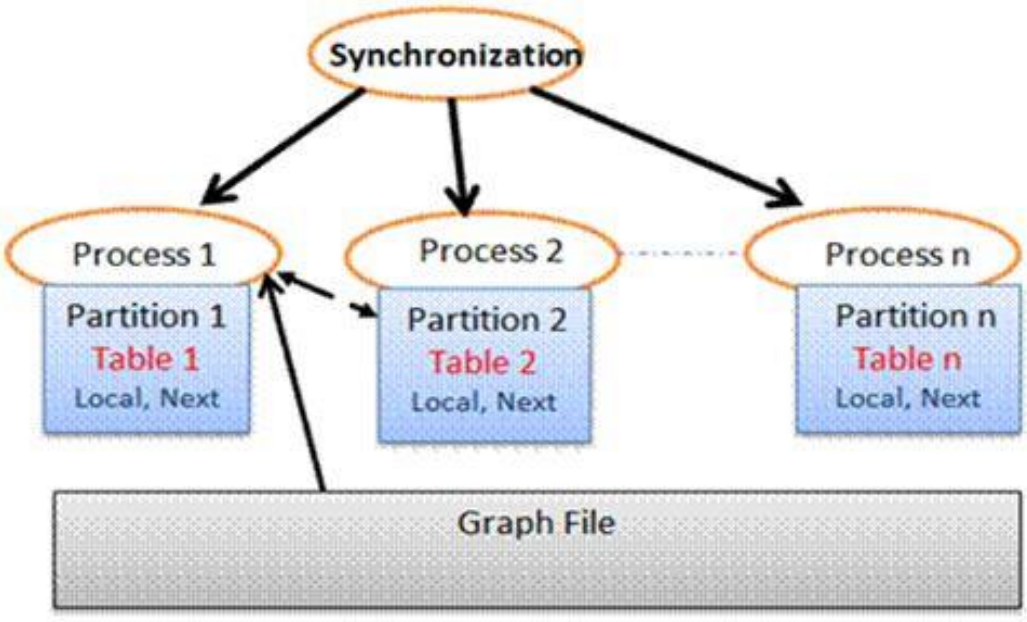


Figure 5-3. Synchronization of distributed binary graph files

Accurate and efficient computation of the CostRank score for a large network has to be addressed in order to secure significant assists among increasing numbers of devices and hosts in the network. Implementing the CostRank calculations in a parallel environment opens up several possibilities in partitioning the data and load balancing the data. During the implementation of partitions with load balance, the binary graph file divided into a relevant uniform part and circulated to different processors participating in rank calculating. This move leads to efficient use of all processors and this improved the overall quality of computing.

Three different methods are deliberated for partitioning, taking from [Hendrickson, Bruce, and Tamara G. Kolda, 2000] [Trifunovic, Aleksandar, and William J. Knottenbelt, 2004], to state matrix among the processors thus:

1-Divide the matrix using a row-wise distribution

2-Divide the matrix using a column-wise distribution

3-Divide the matrix as a 2D grid.

The method preferred for the computations is the row-wise partitioning as multiply the CostRanksrc into the rows of graph file to get the CostRankdest is needed in place of shown in Figure 5-2.

The cluster of computing technology is used. The first step is to equally partitioned the binary graph file G into β files (where P=number of processors), named later BG_i, where $0 \leq i < P$, and allocated each partition to a PC processor. Allocated M_i bytes distributed in the main memory for each processor to do a portion of CostRank calculations, having P n entries to represent the portion of the corresponding $CostRank_{src}$ vector, and create synchronization File Si, storing two kinds of destination state IDs and their $CostRank_{dest}$ scores. This file is called a "synch," to represent the destination rank vector adapted from the serial CostRank algorithm. The algorithm is first partitioned file BG into β portions BG_0, BG_1, .. ,BG_{P-1}, such that each BG_i started from:

$\frac{i*M}{P}$ to $\frac{(i+1)*M}{P}$.

The distribution of attack graph file, as described above, will end to have a specific portion of graph file for each processor. Arrays of $CostRank_{dest,\ i,}$ out-vector, and $CostRank_{src,\ i}$ are created for each PC. The $CostRank_{dest,\ i}$ array and the out-vector array contains only the consistent MP entries, while the $CostRank_{src,\ i}$ array comprises a different set of values needed in CostRank calculations for each iteration, such as total number of states, the number of links etc. The parallel version of the CostRank computation is shown in Figure 5-3 and Figure 5-4. Suppose $M\beta$ receives an update message from some foreign document, and suppose X_{MP1} and X_{MP2} are two documents local to X_{MP}, both of which are out linked by X_{MP}, then one method of update would be to simply increment the CostRanks of X_{Mp1} and X_{MP2} by w(v, u) ← h(v, u)/ sum ×(update message received from X_{MB}). If the absolute change in CostRank were to exceed epsilon (error threshold); then X_{MP1} would send its own local update message to $X_{MP2,}$ This message would trigger a further update in the CostRank of X_{MP2}. In sum, the idea would be to set up a local messaging

analogue between foreign linked documents. Please note, the variance among successive CostRank of a specific document is cast off to calculate the convergence, once the absolute change drops under epsilon (an error threshold), no additional change will bring up to date for that document.

```
/* initialize */
Set all CostRanks to an initial value;
At time = 0;
Concurrently on all hosts:
for all documents in this host {
compute  newrank based on inlinks push newrank into Costrank_dest;
Residual = abs (Costrank_dest - Costrank_src);
if (Residual > Δ) {
if outlink is local
add ( Costrank_dest, update) pair to local update queue
else
send (Costrank_dest, update) message to appropriate foreign host. }
}
/* every host listens for CostRank update messages */
while CostRank update message received {
recompute newrank based on message received {
Residual = abs(Costrank_dest - Costrank_src);
if (Residual > Δ) {
if outlink is local
add ( Costrank_dest, update) pair to local update queue
else
send (Costrank_dest, update) message to appropriate foreign host.
}
}
```

Figure 5-4. Parallel CostRank Algorithm

Figure 5-5 shows the parallel CostRank computation during an iteration using 30 processors and an extra 2 PCs to perform synchronizations of Synch files between processors.

Please note, that the residual calculation is repeated in Figure 5-4, for the purpose of the clarity. Practically the author used a residual function for the computation.

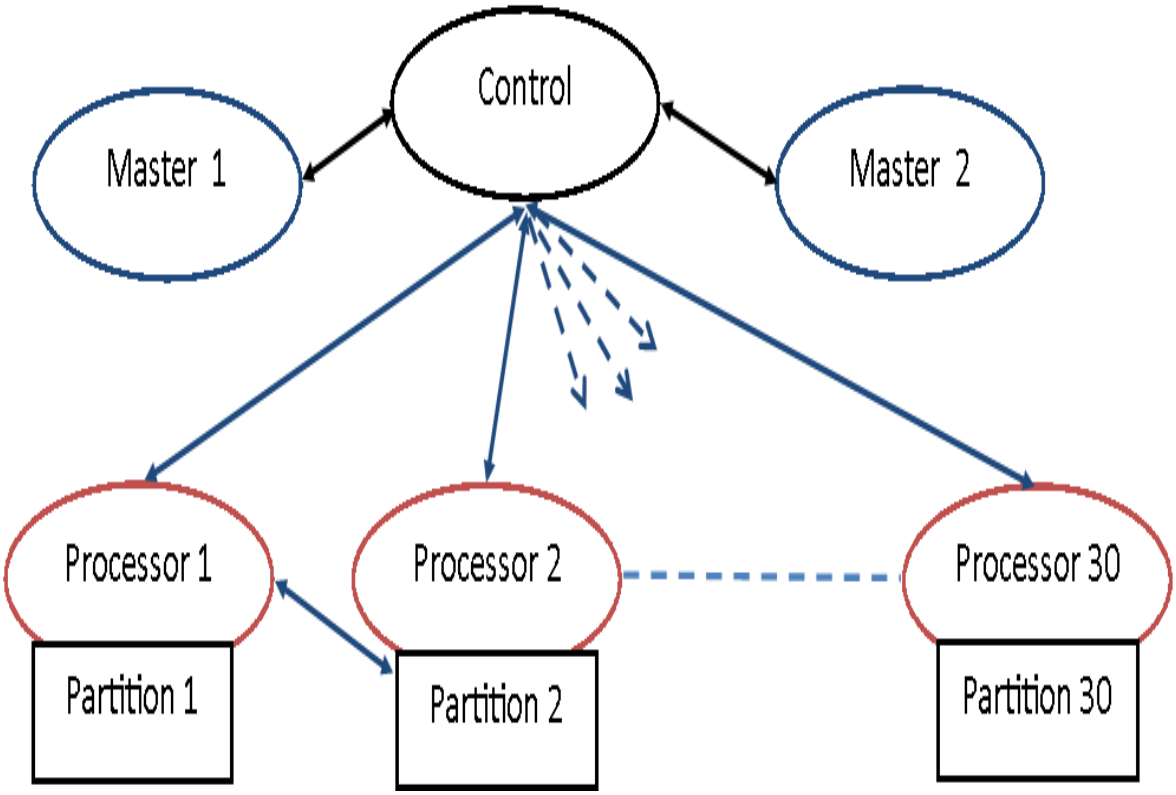


Figure 5-5. Lab Topology Structure of parallel CostRank algorithm

5.4 IMPLEMENTATION OF PARALLEL COSTRANK ALGORITHM AND RESULTS

5.4.1 Lab Topology Setup

Experiments are performed on a cluster of 32 computers, each having Pentium Core2Duo 2.54 GHz processor, 2GB RAM, 250GB Hard Disk interconnected with 100 MB/s Ethernet LAN, running the Linux operating system. The research page at Stanford University been modified. It describes the link structure of the stanford.edu-domain from a September 2002 collection and contains 281903 pages with about 2.3 million links. Stored in Mat lab sparse format, including the cost values and other fields in the graph file it takes up 74.6MB in CRS format. Variable Si corresponds to the synch file, Mi to the partitioned attack graph file, and Vi to the portion of source rank vector in memory existing at processor P. Throughout iterations, the following process occurred: first, computing a new rank based on the corresponding part of graph file Mi then push the calculated values into $CostRank_{dest}$ vector. These processes will continue until the residual value exceeds epsilon. The calculated rank $CostRank_{dest}$ is being added to the local updated queue if the out link is local, or if the out link belongs to a foreign host, it is sent to the master host.

In this thesis, the evaluation of the parallel CostRank algorithm is grounded on the performance and scalability. The performance metrics are based on the total time spent in the CostRank iterations, Speed up time by running Parallel CostRank over the serial CostRank algorithm and the efficiency of CostRank algorithm according to the problem size.

In this segment, a visible logical dialogue has provided in terms the following metrics: I/O rate, Memory Norm, Synchronization rate, Execution times, Speed up, the charge per unit of parallel CostRank algorithms, Efficiency of parallel CostRank algorithm and the Load Balancing.

5.4.2 Input/ output (I/O) rate

In the operation of parallel CostRank calculation, each CPU will perform the following I/O operations:

1- Read an allocated part of an attack graph file |BGi|.

2- Calculate the $CostRank_{Dest}$ rank values |Qi|.

3- The synch file is to save on the disk and re-take it from the master computer for synchronisations. 2x |Si|

Therefore the total read-write I/O rate will be:

$$R_{i/o}parallel = \sum_{0 \le i \prec P} |BGi| + 2|Si| + Qi$$

where Qi = local update queue. At this time, the summation of all P partitions of BGi is equal to the binary Attack graph file BG, and the outline of all synch files Si is S.

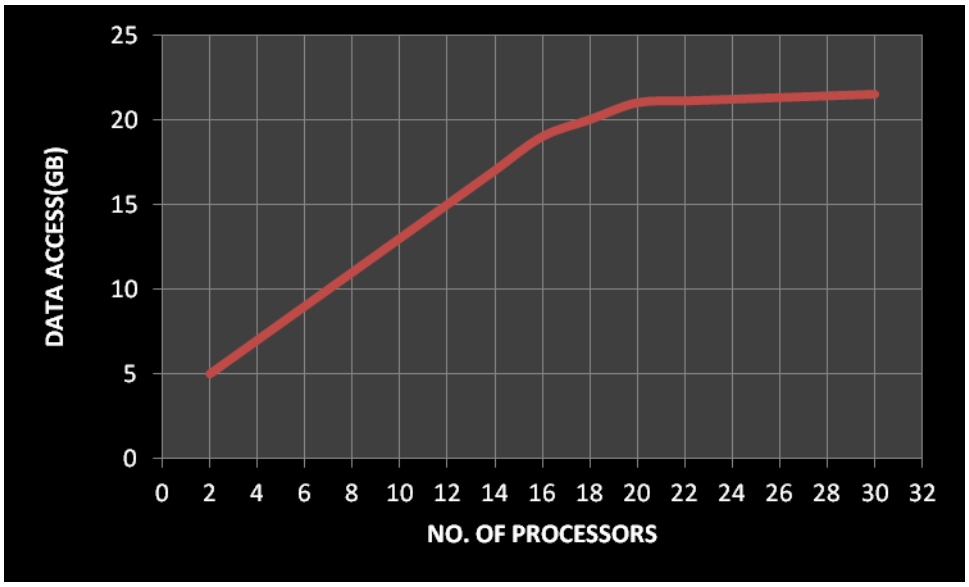


Figure 5-6. I/O rate during iterations and data transport

The amount of data transfer rate and level of iterations in terms of the communications rate represent the efficiency of I/O devices and the bandwidth

for each processor node, and on the start-up latency for the thorough transfer. Following to the analysis above, the I/O cost metric of data access is enlarged by the aspect of 2 x Si and local update queue Qi, which explains the linear increase of I/O with the number of processors up to 20 processors, as seen in Figure 5-6. The overhead of the processors is reduced as the problem size divided using a load balancing techniques, which makes the data access nearly steady after the 20 processors.

5.4.3 Memory Norm

All processors must assign fixed segments of memory to fit an array of $CostRank_{src}$ Mi. For that reason, the overall memory usage is:

$$C_{Mem}parallel = | Mi | = M$$

Since the source rank vector V is divided into β segments, the shape of all segments Vi will be equal to the size of the source rank vector V. Figure 5-7 shows the memory usage per processor during computation.

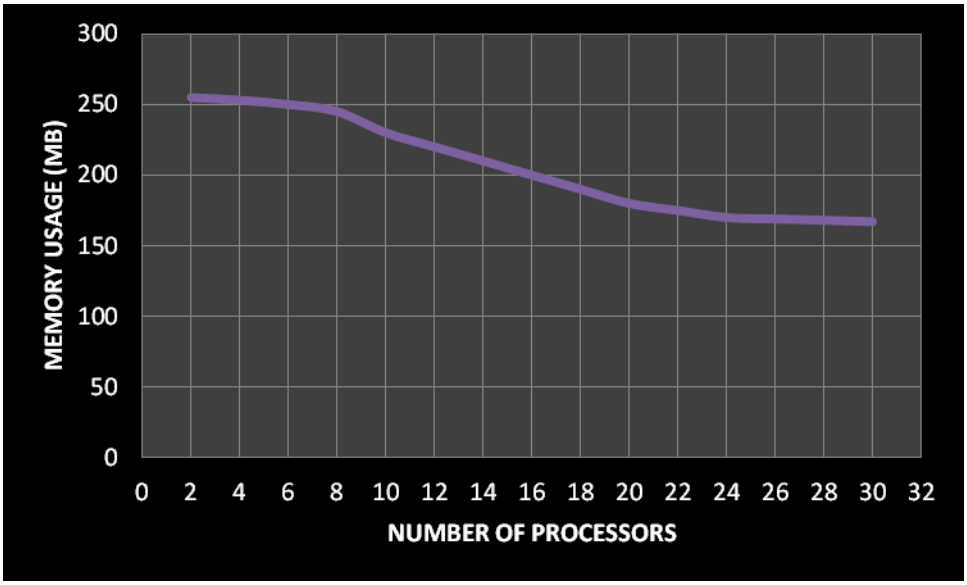


Figure 5-7. Memory usage per processors during computation

As seen in Figure 5-7, there is a high score of memory usage when only two processors are used. This is due to the extra storage needed for synchronized files. When the number of processors is enlarged along with accurate distribution of tasks according to the load balancing techniques in Section 5.4.9, then the total memory usage decreases, and tends to be stable after 20 processors, which related that to the scalability of 2.3 M link structure (please refer to Section 5.5) as shown in the Figure 5-7.

5.4.4 Synchronization rate

As the Synch file Si contains state ID and CostRank value, the size of Synch file Si= 2 * Vi. The synchronization rate among processors is:

$$C_{Synch}parallel = \sum_{0 \le i \prec \beta} |2*(|Si|/2)+|Mi|$$

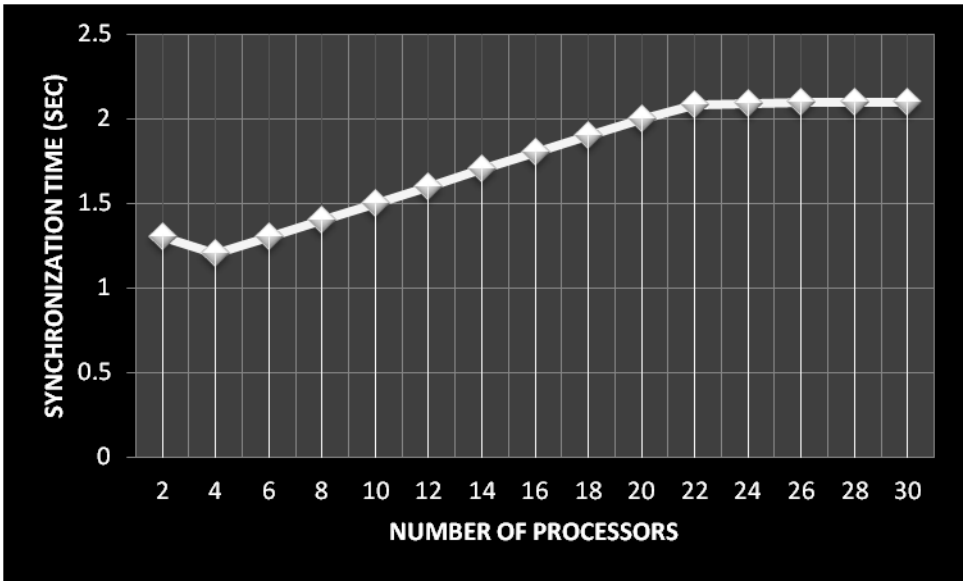


Figure 5-8. Synchronization time vs. number of processors

The local CostRank scores have to be synchronized among the ten processors to obtain the final rank to be used in the next iteration step. Figure 5-8 shows the synchronization time per number of processors. Each processor needs to receive an updated file from the neighbour host through the master (synchronize) machine, process the update, and then send the updated file to the next neighbour host through the master machine. The two master machines need to replicate the data between them. When the CostRank calculations need to get parameter values from another processor, then communication time overhead must be considered. On the other hand, when the master PC has to wait for synchronization from other machines, then the synchronization overhead slightly increases due to waiting time, as the total number of processors increases. Whereas the percentage of communication overhead as you see in Figure 5-9 grows with the number of processors and tends to be stable after 20 processors, due to the scalability of the 2.3 M link structure. Time devoted to communication and synchronization with other processors is calculated using the following formula:

$$Tq = \beta * T_{Parallel} - T_{sequential}$$

$T_{Parallel}$ represent a computation time for a CostRank per processor.

β = processor number.

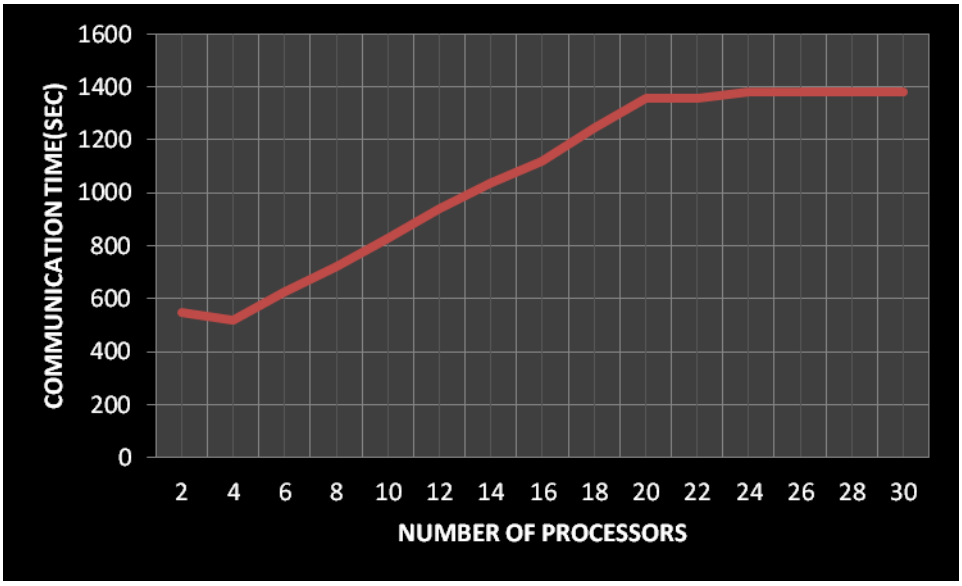


Figure 5-9. Communication time vs. processor number

Figure 5-10 demonstrates the standard of residual errors increases when the synchronization interval increases and when the quantity of processors participating in the calculation is enlarged (in this work, two, four and eight processors are used to calculate the average residual error). The average residual error calculated from synchronizing the local rank scores after every 5 consecutive steps are less than 0.020. This value is used in this research purpose. Thus, the synchronization interval is set to five to be reported in experimental results.

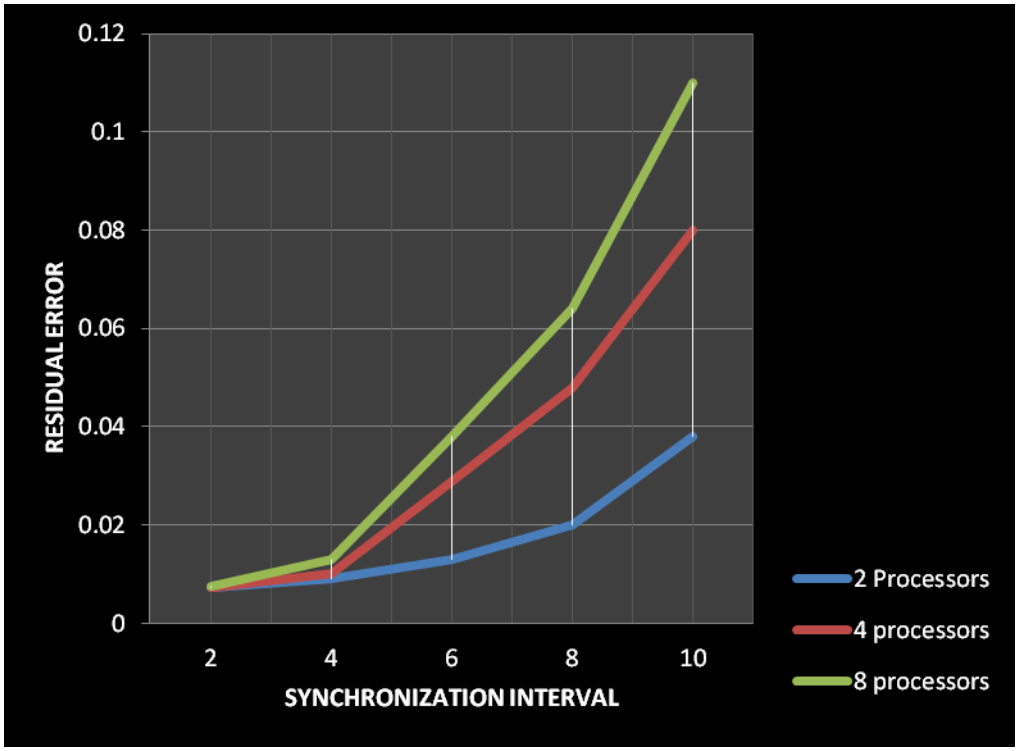


Figure 5-10. Residual Error vs. synchronization interval

5.4.5 Execution times

The execution times should be in theory, equivalent to the serial execution time divided by processor numbers, given the assumptions of serial execution time and the network transfer between hosts and constant computational time per processor. Let $T_{parallel}i$ be the computation time for a CostRank per processor i, $Synch_t$ is the time to synchronize synch files. The execution time can then be estimated as:

$$T_{parallel} = \sum_{0 \le i \prec P} | T_{Parallel}i | + 2 | Synch_t |$$

As the Serial CostRank execution time is divided into p segments, the outline of all portions $T_{Parallel}$ i will be equal to $T_{sequential}$ /p in addition to two synchronized files received from P_{p-1} and P_{p+1} (Overhead).

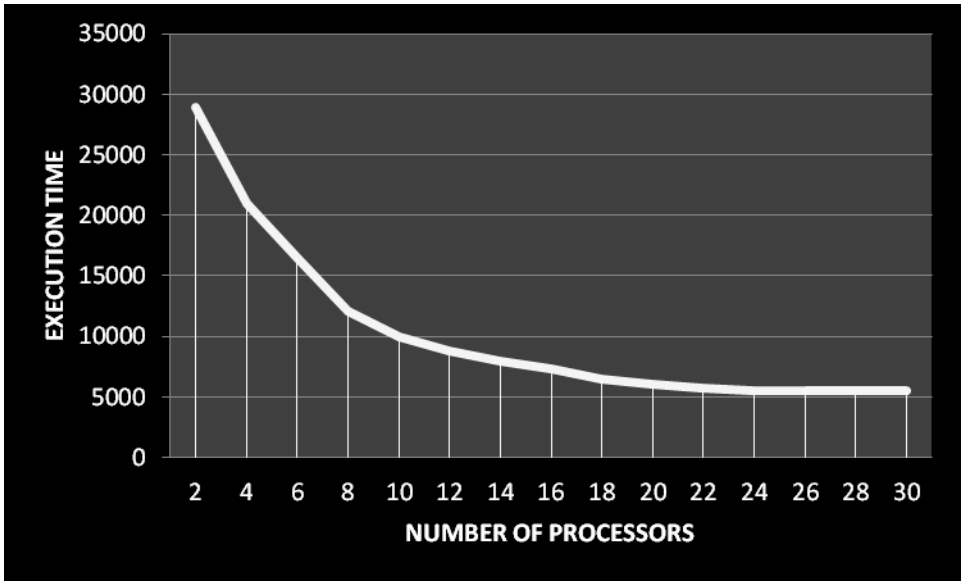


Figure 5-11. Total execution time vs. processor number

The parallel CostRank algorithm and load-balancing scheme permitted us to load and distribute the matrices to the processors without overloading the memory on any particular machine. Figure 5-11 presents the total execution time of the CostRank algorithm against the processors number for a simple distribution (equal number of rows on each processor).

The runtime displays reduced while the processor number increases up to 20 processors, then the trend becomes stable due to the overhead time (more details in Section 5.5), and this behaviour are smoothed with the load-balancing distribution. As the algorithm is so sensitive to the communication pattern in the matrix-vector multiply operation, these peaks correspond to local limits in the communication patterns.

5.4.6 Speedup

Speedup is a measure of the increase in the speed and performance in a parallel algorithm compared with a sequential computing of CostRank algorithm.

SP_p is the proportion of the time of engagement of one processor (using serial algorithm) divided by the total time necessary to calculate on parallel environments using p processors, in order to calculate the CostRanks values.

$$SP_p = \frac{T_{sequential}}{T_{Parallel}}$$

$T_{sequential}$ is the time taken on one processor in sequential algorithms. $T_{Parallel}$ is the total time necessary to compute CostRank on parallel environments using P processors.

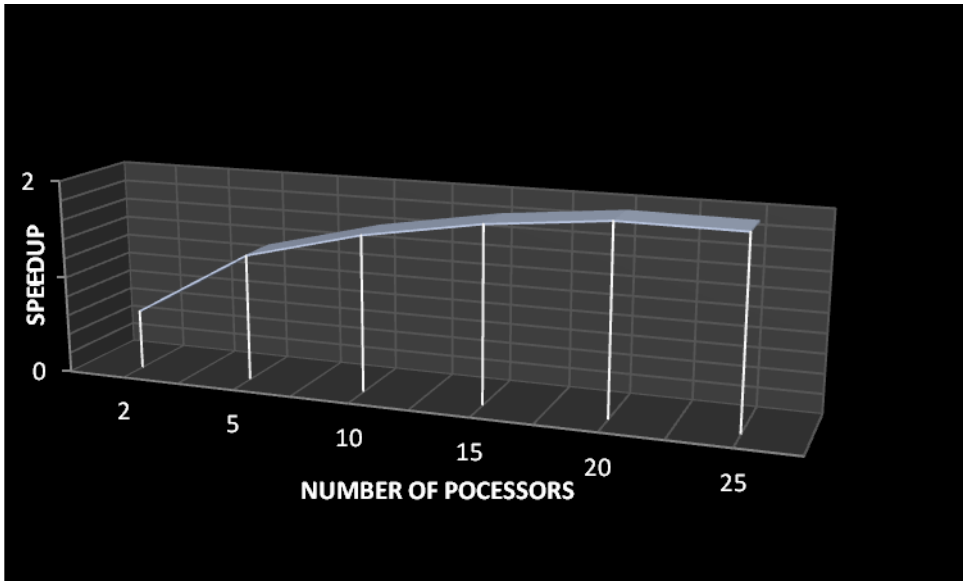


Figure 5-12. Speedup vs. Processor number

Figure 5-12 exposed the speedup of parallel algorithm with regard to processor number. In this graph, speedup is defined to be the ratio of time for the serial CostRank algorithm execution to the time for the parallel algorithm execution. It can be seen that the speedup time is sharply increasing during the experiments. The exceptional case, using only two processors, due to partitioning of the group file, load balancing, and synchronizations and due to the overall loads slumps on each processor. There was a gradual increase up to 20 processors and tends to be stable after 20 processors, which indicate that the 20

processors represent the scalability of 2.3 M link structure according to isoefficiency function. (More details about isoefficiency function in Section 5.5 in this Chapter).

5.4.7 Rate of parallel CostRank algorithm

The total processing time used in analogous parallel algorithms to calculate the CostRank, refer to R_p, where R represents the rate processing in parallel algorithm.

$R_p = \beta *$ Parallel runtime for each processor ($\Pr_p$)

$$R_p = \beta * \sum_{i=1}^{n} \Pr_i$$

Figure 5-13 presents the computation time of the CostRank algorithm against the number of processors. The computation time displays a sharp decline with regard to the increase in processor number up to 20 processors. This is because of load distributions on processors along with using a parallel algorithm to calculate the cost, and then the computation time will become steady due to the scalability and using a fixed problem size.

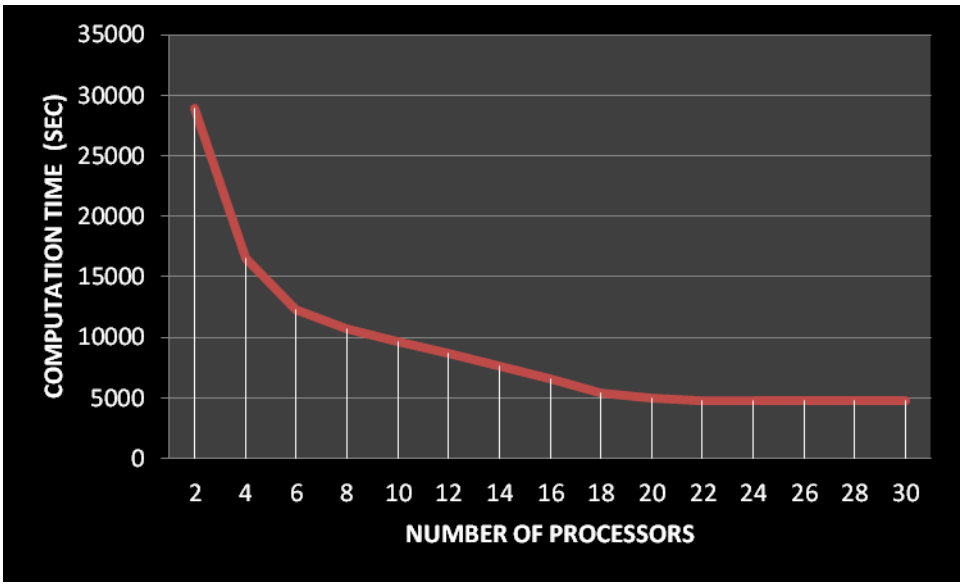


Figure 5-13. Computation time of the CostRank algorithm against the number of processors

5.4.8 Efficiency of parallel CostRank algorithm

E_p is the effective and efficiency time of parallel analogous algorithms, that the processors are performing effective calculations.

$$E_p = \frac{T_{sequential}}{\beta * T_{Parallel}} = \frac{SP_\beta}{\beta}$$

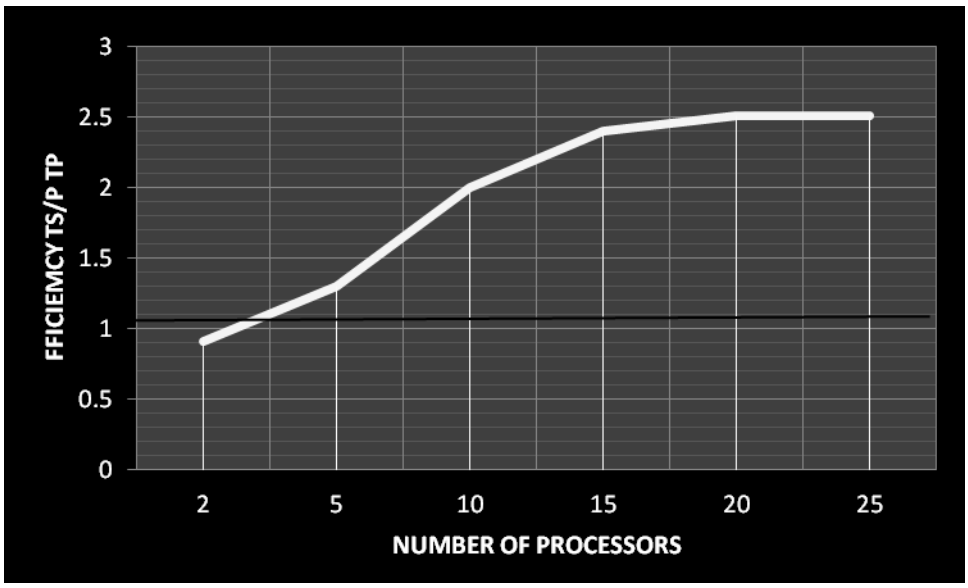


Figure 5-14. Algorithm Efficiency vs. no. of Processors

As the efficiency represents the speedup divided the processor number, as you can see in Figure 5-14, the efficiency increased when the number of processors increased for a fixed problem size, the exception in case of using two processors as the overhead of partitions of links files, load balancing, and synchronizations influence the efficiency. Ultimate efficiency is always one, and anything above one is super-linear and indicates out of bound status. Values less than 1 indicate slow performance and indicate more processors must be added to adjust the

proficiency. The efficiency tends to be stable after 20 processors, which indicate that the 20 processors represent the scalability of 2.3 M link structure according to isoefficiency function.

5.4.9 Load Balancing

The load balancing is equally partitioned and distributed, the binary attack graph file M into β files balanced among processors (P) to accomplish a high deployment of processing usage, where $0 \leq i < P$, and allocated each partition to a PC processor. Allocated M_i is distributed in the main memory for each processor to do a portion of CostRank calculations. Having P n entries to represent the portion of the corresponding $CostRank_{src}$ vector to represent the destination rank vector take on from the Serial CostRank Algorithm, first the attack graph file BG is partitioned into p portions BG_0, BG_1, .. ,BG_{p-1}, such that each BG_i started from: $\frac{i*M}{P}$ to $\frac{(i+1)*M}{P}$.

If p represents the sum of processors used, while M represents the summation of states within the binary attack graph file, then the states are distributed based on the unique state-ID going from $\frac{i*M}{P}$ to $\frac{(i+1)*M}{P}$. Every processor is set to handle a particular value in the range of the function, all states having a state-ID that satisfies a particular processor's value. To achieve this, the Greedy algorithm is implemented with round robin to distribute the tasks according to the number of keys in the table partition to multiple processors, as shown in Figure 5-15. In heterogeneous hardware, it is challenging to ascertain the exact execution time.

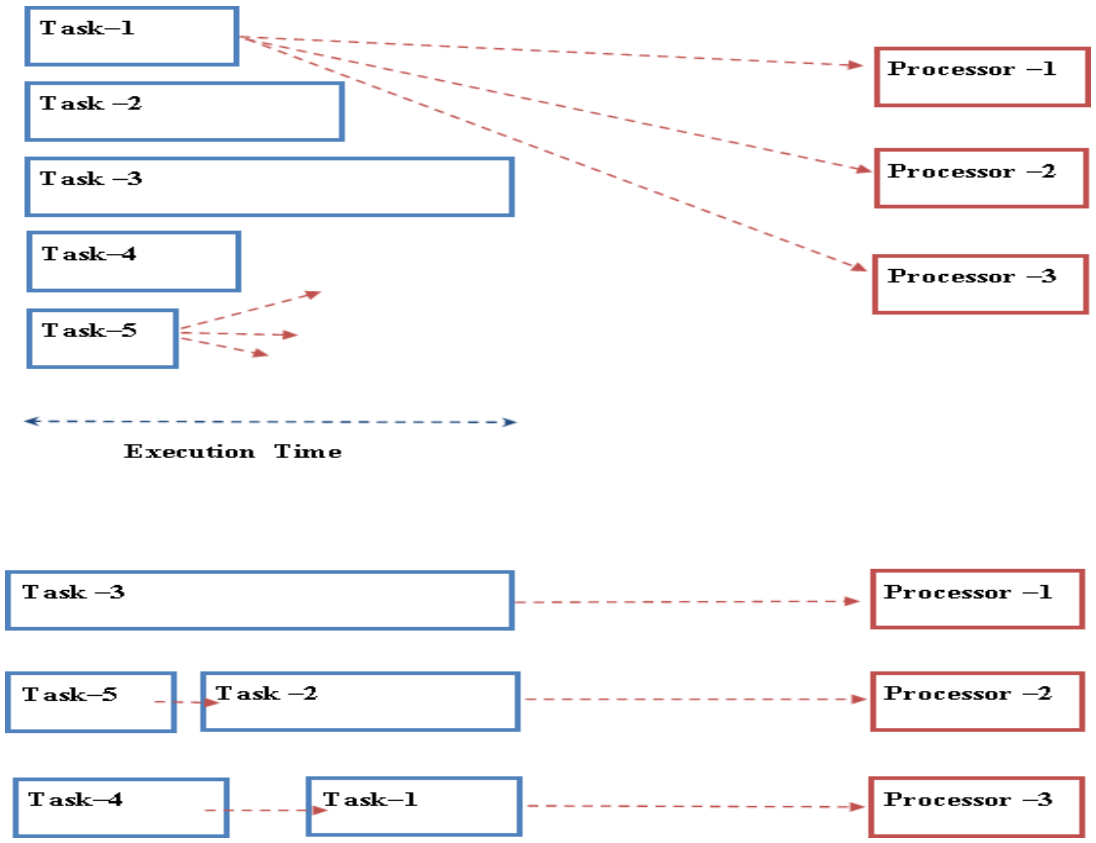


Figure 5-15. Greedy algorithm with round robin

Due to the states' distributions across hosts and processors and due to the use of the Greedy algorithm with the round robin technique shown in Figure 5-16, the load per processor reduces sharply with the number of hosts used during the experiments.

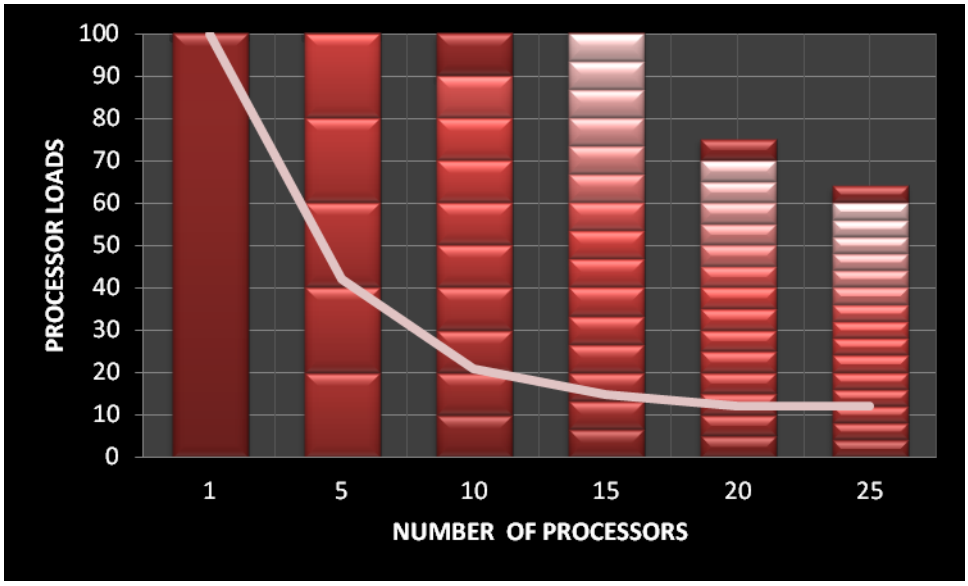


Figure 5-16. Processors loads vs. number of processors

The load balancing represents an important characteristic of any parallel algorithm as load imbalancing that represent uneven processor utilisation and that can contribute to poor efficiency and scalability of the parallel arrangements.

In this work, the binary graph file states' are distributed across processors; and the Greedy algorithm was being utilised with the round robin technique; the cost assigned to transfer part of calculation to another processor is a lot less than the calculation time is required to treat the portion of the graph file. This process was performed using two synchronisation computers to make sure that each CPU receives the calculation part from others.

Based on the above observations, this partitioning scheme yields a satisfactory load balancing. This is a result of the uniform distribution of states within the binary graph file.

5.5 THE SCALABILITY OF PARALLEL COSTRANK ALGORITHM

The evaluation of parallel algorithms based on two metrics, the performance and the scalability.

In section 5.4, an empirical approach to evaluate the parallel CostRank algorithm's performance demonstrated a major reduction in communication overhead and in runtime. Several tests were performed using out-vector files synchronised from the real networked data. The given results are quite encouraging. A partition-centred CostRank algorithm has been produced that can efficiently work in a parallel setting.

In summation, a visible logical dialogue is provided in terms of I/O and synchronization rate, and memory utilisation, speedup gain by using parallel CostRank over the Serial CostRank algorithm, load balancing, processor loads, and efficiency of parallel CostRank algorithms.

To evaluate the CostRank scalability metric, various trials were performed using out-vector files synchronised from the link structure of the stanford.edu-domain from a September 2002 collection and contains 281903 pages with about 2.3 million links, to monitor the constancy of the efficiency when the problem size and the number of the processors increased.

Isoefficiency Metric of Scalability [Grama et al., 1993] is being utilised to evaluate the scalability of CostRank algorithm. When the number of processors is increased using a fixed problem size, the efficiency of the parallel scheme increased to the certain number of processors and then will go down slightly, because of the load balancing techniques, as each processor is not fully utilising the resources.

Using similar principles, when the problem size increased and at the same time, the efficiency increased as the number of processors increased, this indicates bad utilisation of the processing recourses.

The prospect, which maintains the constancy of the efficiency while the problem size and number of processors increased, determines the scalability of the algorithm.

The problem size in this thesis used (P_S) and total overhead time (T_o), which can be calculated using the following formula:

$$T_o = \beta * T_{Parallel} - T_{sequential}$$

The parallel runtime is calculated using the following formula:

$$T_{Parallel} = \frac{P_S + T_0(P_s, \beta)}{\beta}$$

The speedup can be expressed as:

$$SP_p = \frac{T_{sequential}}{T_{Parallel}} = \frac{P_s}{T_{Parallel}}$$

$$SP_P = \frac{P_S.\beta}{P_S + T_o(P_S.\beta)}$$

The efficiency expressed as:

$$E_p = \frac{T_{sequential}}{\beta * T_{Parallel}} = \frac{SP_\beta}{\beta}$$

$$E_p = \frac{P_s}{P_S + T_o(P_S.\beta)}$$

$$E_p = \frac{1}{1 + T_o(P_S.\beta)}$$

The efficiency maintains the constancy (fixed) values between 0 and 1, if $\frac{T_0}{P_S}$, maintains a steady value, then the parallel algorithm can be called scalable. To extrapolate performance from small problems size and small systems to larger problem size on bigger system, the scalability experiments are performed using the following problem size, using 5, 10, 15, 25 processors:

P_S *Links*
0.575 M
1.5 M
1.725 M
2.3 M

The experimental speedup of parallel CostRank execution times is shown in Table 5-1 and in Figure 5-17, as you can see the speedup increased as the number of processors increased. According to Amdahl's law the speed up tends to saturate, in this case study the saturation started at processor 20 for 2.3 M problem size, the maximum speed up for problem size=0.575 M using 5 processors and the minimum for the size=2.3 with 5 processors that was because the different overhead of parallel algorithms and the synchronizations.

In other hand, Speed up time was sharply increasing during the experiments. The special case, when only two processors have utilised, due to the partitioning of the group file, load balancing, and synchronizations. There was a gradual increase in the speedup to 20 processors used and tends to be stable after 20 processors, which indicate that the 20 processors represent the scalability of 2.3 M link structure according to isoefficiency function.

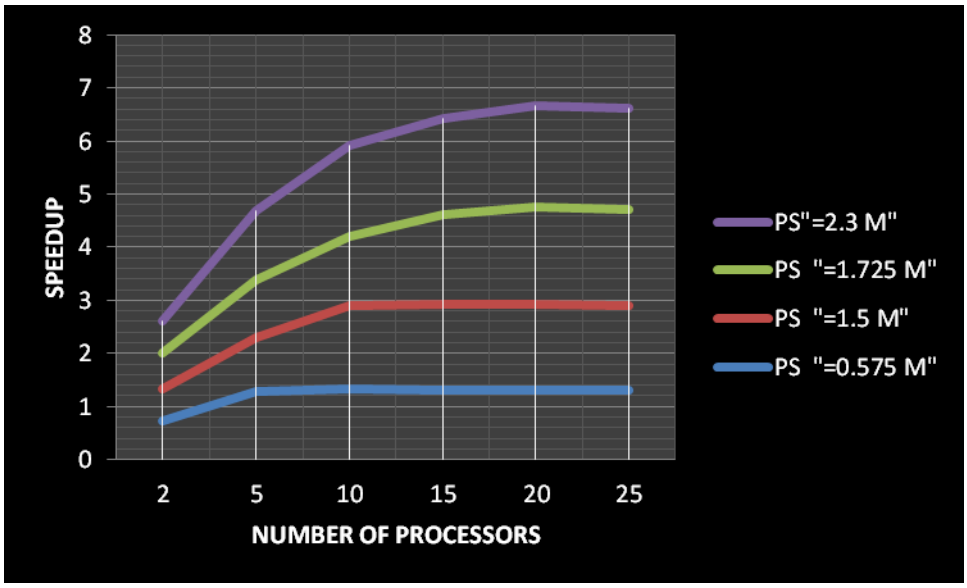


Figure 5-17. Speedup in Parallel CostRank execution times for different problem size

β	P_S				
	0.575 M	1.5 M	1.725 M	2.3 M	
1	1	1	1	1	
5	1.29	1	1.1	1.3	
10	1.32	1.59	1.3	1.7	
15	1.319	1.6	1.7	1.8	
20	1.31	1.61	1.83	1.92	
25	1.3	1.59	1.81	1.91	

Table 5-1. Speedup in parallel CostRank execution times for different problem size.

P	P_S				
	0.575 M	1.5 M	1.725 M	2.3 M	
1	100	100	100	100	
5	24.9	20.2	12.8	9.1	
10	30	25.0	23.1	20.0	
15	27.1	26.1	25.1	24.1	
20	25.33	25.31	25.3	25.11	
25	25.11	25.19	25.15	25.1	

Table 5-2. The efficiency of parallel CostRank execution times for different problem size.

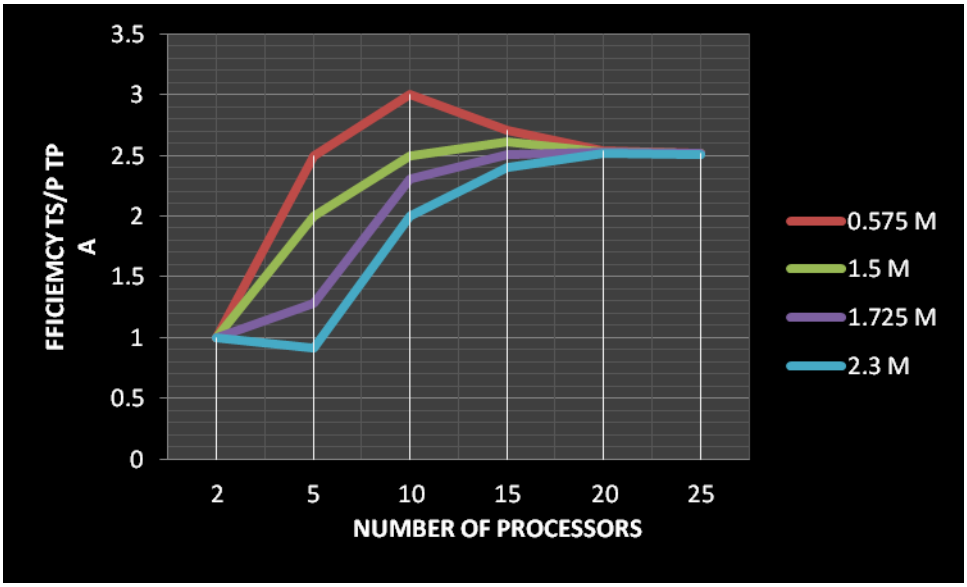


Figure 5-18. Efficiency of Parallel CostRank execution times for different problem size

The problem size in this practical model can be expressed as an isoefficiency function of the processor number in order to keep the efficiency steady.

$$E_p = \frac{1}{1 + T_o(P_S.\beta)/P_S^{'}}$$

$$= \frac{T_o(P_S,\beta)}{P_S^{'}} = \frac{1-E}{E}$$

$$P_S^{'} = \frac{1-E}{E}(T_o(P_S,\beta))$$

As the goal to keep the efficiency steady, in this case $\frac{1-E}{E}$ should be constant=C.

The relationship between the problem size and the overhead of the processors has used to calculate the scalability of parallel CostRank and demonstrated as follows:

$$P_S^{'} = CT_o(P_S,\beta)$$

As you can see in Figure 5-18 and Table 5-2, The efficiency tends to be stable at (0.575 M, 5), (1.5 M1, 10), (725 M, 15), (2.3 M, 20) with the value around 25 which indicate that the 20 processors represent the scalability of 2.3 M link

structure according to isoefficiency function. That the scalability (isoefficiency function) of the parallel CostRank algorithm is $S(P_S) = \Theta(P_S^2 \log P_S)$.

When the problem size is fixed and the figure of the processors increased, the efficiency has to be likely to increase and when the number of processors is set, while the problem size increased, the efficiency has become less efficient. Please notice that Ultimate efficiency is always 1 (scaled to 100 in this practice model).

As expected according to isoefficiency approach, the problem size was increased with the number of the processors to keep a fixed level of efficiency, which indicates good utilisation of the processing resources in term of processors and network specifications setting.

As exemplified above, the isoefficiency function, which has been used to monitor the efficiency to be nearly steady, according to the speedup and the number of processors, as you can view in Figure 5-18.

The existing results of speedup and efficiency of the Stanford link structure show the scalability of using 20 processors as the overheads of each processors reduced.

When more than 20 processors are used, the overhead is increased that means the processors become less involved in process of CostRank calculations as the efficiency tend to be stable and the speedup tend to saturate.

The existing results showed the pattern of scalability of increasing the number of processors according to the problem size.

The algorithm offers an excellent estimate of real parallel CostRank calculations, however, the wall clock time performance metric when the algorithm used different machines and network specifications, reduces convergence time, extensive memory demanding metrics are not considered as they are not in the scope this study and they are suitable to be investigated in future work.

5.6 CONCLUSIONS

In this chapter, a new parallel algorithm is applied to implement CostRank for distributed parallel computers using multiprocessors. To reduce the complexity of the serial CostRank algorithm (as it includes matrix multiplication, the complexity = $\Theta(n^2)$) and to make CostRank calculations are more effective in term of the cost/benefits indication. When the number of hosts in the networks is increased, the serial CostRank algorithms need to read the source CostRank values of different states and store it in a buffer, then read the parameters in the binary graph file header. This has three components are entire number of states; the entire number of links; and the upper limit out degree in the attack graph. A state entry record saves the essential structural data for a state such as state ID, the out-degree, and then forward links (Link 1, Link 2, and Link v) along with corresponding cost for each link (C1, C2, and Cv) respectively. The parallel in-memory Piccolo algorithm is selected to extensively reduce access share state stored in memory and iterations, which include table partitioning (local access), synchronization of distributed table, checkpoint/restore and load balance/ and task scheduling. In the same way, the large scale networks are secured that require fast and reliable computing to calculate the ranking of enormous graphs with thousands of states and millions or links. In this, a focus on a parallel CostRank computational architecture on a cluster of PCs networked for partitioning calculations and two masters, for synchronisations to assess the complexity and scalability of the algorithms. It describes the link structure of the stanford.edu-domain from a September 2002 collection and contains 281903 pages with about 2.3 million links, outcomes in a major reduction in communication overhead and in runtime. To study its efficiency, several tests were performed using out-vector files synchronised from the real networked data.

A parallel CostRank computational architecture has been used along a bunch of PCs networked via 100 MB/s Ethernet LAN, using a cluster of 32 computers built with Pentium Core2Duo 2.54 GHz CPU, 2GB RAM, 250GB Hard Disk

interconnected with 100 MB/s Ethernet LAN, running the Linux operating system. 30 PCs for partitioning calculations and two masters, for synchronisations to assess the performance and scalability of the CostRank algorithms.

An analysis of the performance and scalability of the parallel CostRank algorithm is presented; the empirical approach to evaluate the CostRank algorithm's performance indicates a major reduction in communication overhead and in runtime.

The Total time expended in the CostRank iterations is reduced while the processor number increases up to 20 processors, then the trends become stable due to the overhead time, the problem size and algorithm scalability, in other hand Speed up time were sharply increasing during the experiments. The special case, when only two processors have utilised, due to the partitioning of the group file, load balancing, and synchronizations. There was a gradual increase in the speedup to 20 processors used and tends to be stable after 20 processors, which indicate that the 20 processors represent the scalability of 2.3 M link structure according to isoefficiency function.

The efficiency of CostRank algorithm according to the fixed problem size (2.3 M link structure) increased when the number of processors increased, the exception in case of using two processors as the overhead of partitions of links files, load balancing, synchronisations influence the efficiency.

The efficiency tends to be stable after 20 processors, which indicate that the 20 processors represent the scalability of the 2.3 M link structure.

A visible logical dialogue has provided in terms of I/O and synchronization rate, and memory utilisation, speedup gain by using parallel CostRank over the Serial CostRank algorithm, load balancing, processor loads, and efficiency of parallel CostRank algorithms.

The Isoefficiency function of Scalability [Grama et al., 1993] is used to measure the scalability of CostRank algorithm.

An experiment is performed to investigate the scalability of the Parallel CostRank algorithm. As expected according to isoefficiency approach, the

problem size increased with the number of the processors to keep a fixed level of efficiency, which indicates good utilisation of the processing resources using the experimental network, bandwidth, and platforms setting.
The business benefits of employing a parallel CostRank algorithm for enterprise network are: (i) speed up the process of detecting and mitigating the vulnerabilities in the enterprise network. (ii) As the dynamic approach considers the interactions between the vulnerabilities, that will increase the efficiency of security protections. (iii) The dynamic, parallel CostRank is scalable, and consider as a suitable solution for enormous network systems.

In the next chapter, a novel effective countermeasures solution built up against network attacks using cost-centric model by identifying the set of top network risks and host and application vulnerabilities and countermeasures. In the DCAT model, attack detection and mitigation permitted not just at the leaf node, but at the transiting nodes as well. The effects of incorporating countermeasures and attacks in the DCAT are studied, using the DVSS framework and the Nessus scanner for vulnerability detections, to construct both dynamic Cost Centric Attack Trees (CCAT) and Dynamic Countermeasures Attack Trees (DCAT).
A business case study is presented, using security investment analysis based on Daniel Bernoulli using Expected Utility Theory to forecast the expected benefits and the net expected benefit and Internal Net Return (IRR) of the dynamic security investment approach against the standard approaches of mitigations.

CHAPTER 6. DYNAMIC COST CENTRIC RISK MITIGATION MODEL

6.1 INTRODUCTION

This chapter identifies and clarifies the cost centric risk mitigation model, a set of top network risks and vulnerabilities (please see appendix A) are used to calculate the dynamic cost using a dynamic cost framework and DVSS explained in Chapter-3, and the countermeasures that are appropriate to address each vulnerability. In order to develop an effective dynamic countermeasures model against network attacks.

Developing a countermeasures and mitigation model is a complex and challenging goal for security professionals, system administrators, and information security researchers, as it is extremely significant to secure a network system.

The wide variety of computer hardware; the complication of operating systems; the diversity of potential vulnerabilities; and the extraordinary skills of many attackers combine to create a problem, which is exceptionally hard to tackle computationally and from a human perspective to be able accurately identify, classify, and provide a better countermeasures model against the many types of malicious threats. That is emphasising the need to produce a dynamic, attack and a countermeasure model [Granadillo, Gustavo and Daniel Gonzalez, 2013]. the attack modelling should identify the significant attacker's goals and must pursue a systematic plan of attack in terms of vulnerabilities identified by the system administrators and not by the attackers during and after vulnerabilities exploitation.

Using security a systematic attack model should ensure a full understanding of network systems and software applications used in the system as a total understanding of the systems, will help to identify the vulnerabilities that exist, and affect the important assets and help to rank them according to the severity, thus finding appropriate mitigations and countermeasures.

The systematic security approaches will also help to find different types of weaknesses in configurations and trust levels between different domains in order to provide effective protection against various malicious attacks. The acknowledged vulnerabilities will be examined carefully and analysed using quantitative numerical dynamic risk propositions in terms of integrity, confidentiality, and availability (CIA) of the vulnerability. In this model, identifying the vulnerabilities and threat priority uses a dynamic Cost-Centric Attack Tree (CCAT) approach; threats are prioritised using a CCAT algorithm to determine critical vulnerabilities that enable the attacker to reach their goal. The root of the CCAT represents the attacker goal and each leaf node represents multi-step attack vulnerabilities to move to the next level of attack. CCAT used AND & OR refinement, AND as defined in the general AT [Edge, Kenneth S, 2013], the attacker should compromise all leaf node vulnerabilities to reach the goal or to progress to the next level of attack. While OR refinement means at least one vulnerability should be compromised to reach the goal or to progress to the next level of attack.

Dynamic cost-centric approach is used, covered in chapter-3, to represent the vulnerabilities in the numerical cost values and probabilities to calculate the effective numerical values to reach the attacker goal. Representing CCAT with dynamic cost centric driven from DVSS is a new approach to build effective AT.

Defence tree (DTs) [Bistarelli Stefano et al., 2006] have been developed to investigate the effect of defence mechanisms using measures such as attack cost, security investment cost, weight, return on attack (ROA), and return on investment (ROI) [Roy Arpan et al., 2010]. However, placing defence mechanisms only at the leaf nodes, the corresponding ROI/ROA analysis does not incorporate the probabilities of attack, while the costs and probabilities are addressed in CCAT, DCAT models.

In the Dynamic Countermeasures Attack Trees (DCAT), the attack and countermeasures are combined in one tree using a cost-centric and probability

represented in CAT algorithms, calculating the effective security measures in terms of countermeasure implementations and having the option to apply different mitigation strategies. To achieve optimal security protections, the DCAT implementations the valued assets of the state space model are taken according to the implementations of CostRank algorithms, discussed in chapter-4.

This chapter introduces a novel DCAT pattern, which brings into account attacks and countermeasures. In the DCAT model, attack detection and mitigation are permitted not just at the leaf node, but at the transiting nodes as well. The effects of incorporating countermeasures and attacks in the CAT are examined utilising the case studies presented in chapter 3, using DVSS framework and Nessus scanner for vulnerability detection and both CCAT and DCAT are constructed. The analysis of cost value and probability is performed to reach a specific goal and compare to existing results to evaluate the effectiveness of the approach.

6.2 DYNAMIC COST CENTRIC ATTACK TREES (CCAT)

Attack trees are the method widely used in estimating and assisting the security operations of complex network systems. Bruce Schneier [Schneier, 1999] defines attack trees as follows:

> *Attack trees afford a formal, systematic manner to represent the protection of a system based on different attacks, fundamentally, the representation of an attack against a system in a tree arrangement, within the end represents the origin node, and the leaf nodes represent the different techniques to gain the destination.*

The AT has been used in many of the real world security implementations, such as digital content security protocols and Mobile devices, WAP assisted protocols [Edge, Kenneth S., 2010].

The root in an AT represents the terminal destination of an attacker, the source node is then refined into sub-goals and the sub-goals are further processed to reach the leaf node which represents the specific attacker's action.

In the attack trees there are two kinds of refinement:
1- Conjunction: refined sub-goals are satisfied if all children are true.
2- Disjunction: refined sub-goals are satisfied if any of the children is true.

Figure-6-1 shows a simple attack tree, where in order for the attacker to reach the goal node, he should satisfy (Exploit-1 or Exploit-2), and (Exploit -3 and Exploit -4). In the DCCAT model, the following levels are demonstrated:

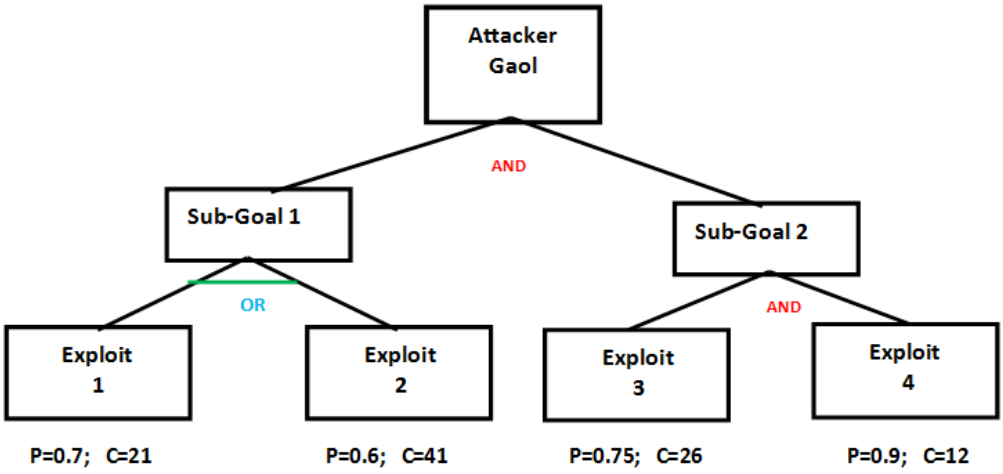


Figure 6-1. Example of an Attack Tree

1. The root of the tree at first level represents the valued assets to be protected. The vulnerability details are obtained by implementing the dynamic CostRank Algorithms.
2. The intermediate level represents states as security properties that enable the attacker to reach the goal or to progress to the next level of the attack.
3. The leaf of the tree, at the third level, represents a different attacker exploit.

These three levels can be repeated in the CCAT as you can see in Figure 6-7 and 6-8.

Fault and Attack trees have been practiced to resolve many computer-based problems including computer and network security [G. Helmer et al., 2002] Different metrics are employed and put to every node in the attack tree in order to compare the different numerical values with other nodes in the tree and to evaluate the cost to complete a specific attack. Different metrics and scales are used in different types of AT, beside the probability of success or failure, in different manners according to a systems requirement and applications.

There are different types of AT based on the metrics used, for example, vulnerability trees, which use vulnerability metrics; defence trees and threat trees.

Formal definitions of a CCAT are delivered and mathematical representations of AND & OR adapted and modified from [Edge, Kenneth S., 2010] and [Mauw, Sjouke, and Martijn Oostdijk, 2006] as follows:

	AND- refinement	**OR- refinement**
Probability	$\prod_{k=1}^{n} P_k$	$1-(\prod_{k=1}^{n}(1-P_k)$
Cost	$Min(C)-\sum_{k=1}^{n-1} C_k /100$	$Max(C)-((\sum_{k=1}^{n-1} p_k * C_k)/(\sum_{k=1}^{n} p_k /100))$
P=Probability, C=Cost.		

As an example, to implement the above formulae, referring to Figure 6-1, the following random parameters are applied:

$P_{Exploit1} = 0.7$

$C_{Exploit1} = 21$

$P_{Exploit2} = 0.6$

$C_{Exploit2} = 41$

$P_{Exploit3} = 0.75$

$C_{Exploit3} = 26$

$P_{Exploit4} = 0.9$

$C_{Exploit4} = 12$

In sub-goal-1, where the nodes connected using OR refinement:

$$P_{Sub-goal1} = 1 - (\prod_{k=1}^{n} (1 - P_k)$$

$P_{Sub-goal1} = 1\text{-}((1\text{-}0.7)*(1\text{-}0.6))$

$P_{Sub-goal1} = 0.88.$

$$C_{sub-goal1} = Max(C) - ((\sum_{k=1}^{n-1} p_k * C_k) / (\sum_{k=1}^{n} p_k / 100))$$

$C_{sub-goal1} = 41\text{-}((21*0.7)/(1.3*100))$

$C_{sub-goal1} = 40.917.$

In the sub-goal-2, which the nodes connected, using AND refinement:

$$P_{Sub-goal2} = \prod_{k=1}^{n} P_k$$

$P_{Sub-goal2} = 0.75*0.9 = 0.675.$

$$C_{sub-goal2} = Min(C) - \sum_{k=1}^{n-1} C_k / 100$$

$C_{sub-goal2} = 12 - (26/100).$

$C_{sub-goal2} = 11.47$

In the final attacker goal, which the nodes connected, using AND refinement:

$$P_{goal} = \prod_{k=1}^{n} P_k$$

$P_{goal} = 0.88*0.675 = 0.594.$

$$C_{goal} = Min(C\ ) - \sum_{k=1}^{n-1} C_k / 100$$

$C_{goal} = 11.06$

As shown in Figure 6-2, x+1 factors must be calculated to obtain the final attacker goal cost and probability, as x+1 represent the parent node. Cs_x and Ps_x represent the successful attack probability and cost for the node x.

In order to calculate the x+1 cost and probability, the path from the root to the leaves in the tree will be navigated twice for each run.

In this case, the sets Ps_x, Cs_x where x=1, 2, 3 , … , n represent the success probability and cost for all the node n and they are self-determining from each other according to OR/AND refinement.

The OR procedure (Psk, Csk) represents the procedure to calculate the cost and the probability if the nodes are OR refinement with root or parent, while ANDprocedure (Psk, Csk) represents the procedure to calculate the cost and the probability if the nodes are AND refinement with root according to the Edge et al formulae.

A basic algorithm to make an AT including the calculated metrics up to the root node are shown in Figure 6-2.

Input: AT elements with leaf nodes set L= {L_1, L_2 ,…, L_n}
Success attack probability s $\in S_n$, Success attack cost c $\in$ Cn
Output: The Cost and probability of the root node (Attacker goal)
Begin
Assign initial values (Probability and Cost) for all leaf nodes
For $\forall L_x \in \{L_1, L_2, \ldots, L_n\}$ Do
Ps: =0, Cs: =0
End For
Assign metrics for all leaf nodes
For x: =*1* to *n* Do
If the node.type="OR"
$Ps_x, Cs_x \leftarrow$ ORprocedure (Ps_k, Cs_k)

$$Ps_k := 100 - (\prod_{k=1}^{n}(1 - P_k)$$

$$Cs_k := Max(Cs_k) - ((\sum_{k=1}^{ns-1} Ps_k * C_k) / \sum_{k=1}^{ns}(Ps_k / 100))$$

End If

If the node.type="AND"
$Ps_x, Cs_x \leftarrow$ ANDprocedure (Ps_k, Cs_k)

$$Ps_k := \prod_{k=1}^{ns} Ps_k$$

$$Cs_k := Min(Cs_k) - \sum_{k=1}^{ns-1} Cs_k / 100$$

End If
End For
End

Figure 6-2. Algorithm to build a Cost Centric Attack Tree

6.3 DYNAMIC COUNTERMEASURES ATTACK TREES (DCAT)

As mentioned earlier, there are many shapes and types of AT, mainly dependent on the metrics used to build and formulate the trees. DCAT is an annex of the AT symbolising attacks against a specific asset and the countermeasures required to mitigate different vulnerabilities to achieve high degrees of protection and security. In the DCAT model, attack detection and mitigation are permitted not just at the leaf node, but at the transiting nodes as well.

The effects of incorporating countermeasures and attacks in the DCAT, using the DVSS framework and the Nessus scanner for vulnerability detections are examined to construct both attack trees and DCAT. The idea of mixing countermeasures into attack trees, and more generally into directed acyclic graphs, using dynamic cost centric driven from DVSS and probability is a new approach. The main shift between CCAT and a DCAT is the CCAT represents the attacker actions using dynamic cost centric values and probability, while DCAT added a set of vulnerability mitigations to reduce the possible damage by an attacker to a critical asset in terms of protection. The root of the DCAT is linked to a critical, valuable asset that the system administrators need to secure and give special consideration. The leaf nodes in the DCAT represent either multi-step sub-goals, which enable the attacker to compromise the root and might damage the critical asset or represent a set of mitigations as countermeasures of vulnerabilities to reduce the impact and change the total dynamic cost and probability to reach the goal. Non-leaf nodes could be represented in two different techniques as shown in Figure 6-3. For OR &AND nodes for an attack OR^{A} and AND^{A}, is utilised and for countermeasures OR^{C} and AND^{C} is applied. AND^{A} means the attacker needs to compromise all leaf node vulnerabilities (let consider two nodes with M, N costs) to reach the goal or to progress to the next level of attack the cost will be <Maximum (M, N). While AND^{C} refinement means at least one vulnerability should be compromised to reach the goal or to progress to the next level of attack OR^{A} ≤Minimum (M, N). OR^{A} refinement means at least one vulnerability should be

compromised to reach the goal or to progress to the next level of attack OR^A ≤Minimum (M, N), while OR^C means the attacker should compromise all leaf node vulnerabilities to reach the goal or to progress to the next level of attack. The DCAT can be described as CCAT enhanced by adding a set of mitigations to each leaf node with vulnerability. To represent optimal manners of vulnerability mitigations in DCAT to calculate the effective cost after implementing the mitigations, the system administrator will calculate the effective cost of possible mitigations in an attack scenario using CAT algorithms to reduce the total impact of a specific vulnerability.

Mitigation(s) for a single vulnerability might be able to stop one multi-step attack or more. On the other hand multiple mitigations for multiple vulnerabilities may be able to prevent or stop one malicious attack, but in some cases, multiple mitigations fails to stop any attack, as preventing an attack is depends on the structures and distributions of the vulnerabilities and mitigations. The valued assets are obtained of the state space model according to the implementations of a dynamic CostRank algorithm; in standard security mechanisms, vulnerabilities are acknowledged using a vulnerability scanner such as the Nessus scanner. Vulnerabilities forming procedures and vulnerability analysis using DCAT are essential to systematically mitigate the vulnerabilities and find possible solutions for system protection. Calculating the effective security standards in terms of countermeasures implementations and having the option to apply different mitigations to reach optimal security protections is the essential process to effectively protect and secure system in a cost/benefit approach.

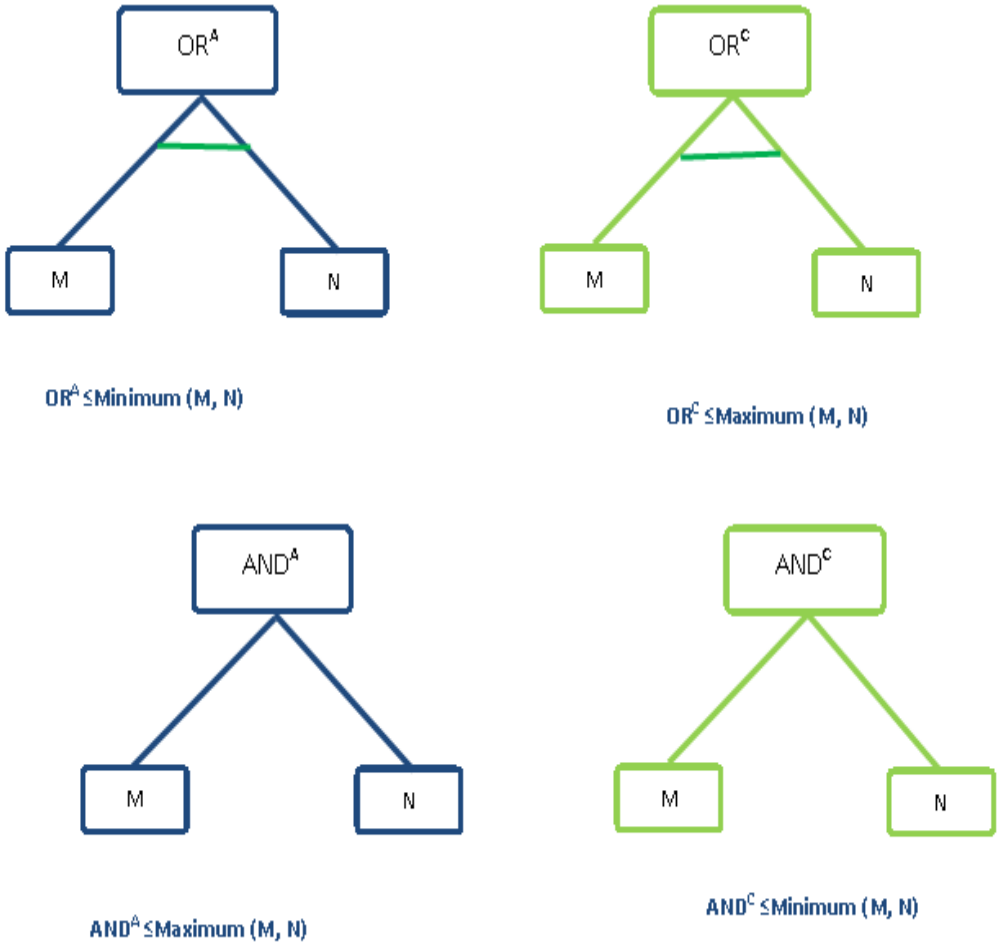


Figure 6-3. Attack trees and Dynamic Countermeasures Attack Trees

6.4 FORMAL REPRESENTATION

In order to represent the Dynamic Countermeasures Attack Tree (DCAT), uses an unranked function F with Domain D and range of functions R, N represents the number of states in a specific space state.

(fs) $S \in \mathrm{N}$ where $f_S : D^s \rightarrow R$ for $s \in S$

Definition 5-1

A CAT–symbol is a pair $\sum(P,F)$ such that

P = {a, c} is a set of attacks (a) and countermeasures(c), (set $\bar{a} = c$ and $\vec{c} = a$),

$$F = \{(OR_S^A)_{S\in N}, \{AND_S^A)_{S\in N}, (OR_S^C)_{S\in N}, (AND_S^C)_{S\in N}, \Re^A, \Re^C\} \bigcup \beta^A \bigcup \beta^C$$

is a set of functions of operations, with mapping type $F \rightarrow P^* \times P$, which expresses the type of each function symbol, as follows:

$\bigcup$ Denoted as a disjoint set union.

For every S ∈ N, $\Re$ represents a tree with root node only; β represent a non-decomposed node.

Definition 5-2

The set of all DCAT terms is denoted by:

$\prod_{\Sigma}$ Since $s \in (a,c)$ defined by $\prod_{\sum}^{C}$ the set of all countermeasures and mitigations, $\prod_{\sum}^{A}$ the set of all attacks. The set of attackers and countermeasures can be represented as follows: $\prod_{\sum} = \prod_{\sum}^{A} \cup \prod_{\sum}^{C}$

The CAT terms of the Attacker and Definer (Countermeasures, Mitigation) type are built exclusively from the basic actions of the attacker $\overset{A}{OR}, \overset{A}{AND}$ and basic action of the definer as shown below.

Definition 5-3

A Dynamic Countermeasure Attack Tree (DCAT) is a finite CAT tree T of a set of symbols

$$\underset{T}{F} = \overset{A}{\beta} \cdot \bigcup \cdot \overset{C}{\beta} \cdot \bigcup \cdot \{OR^A, AND^A, OR^C, AND^C\}$$

Alongside function, which define the type tree as Attack tree or countermeasures tree with OR, AND refinement.

$$\Psi : type(T) \rightarrow \{A, C\}$$

which satisfies the following two conditions for every $\Psi \in Type(T)$

If there exist $i \in \mathrm{N}/\{0\}$ such that $p_i \in type(T)$ and $\Psi_{pi} = \Psi_p$

then

$$T(p) = \begin{bmatrix} OR^A & AND^A \\ OR^C & AND^C \end{bmatrix}$$

Otherwise

$$\mathrm{T\ (p)} = \begin{Vmatrix} B^A \\ B^C \end{Vmatrix} \quad if \quad \Psi_{pi} \neq \Psi_p$$

The function Ψ allows us to differentiate between composed nodes(nodes with child's) and a non-decomposed node(nodes without child's). The value of Ψ_ϕ determining the acts as an attacker or defender to use (AND, OR) or β .

By comparing the value of Ψ applied to the root with values applied to the children p_i, can decide which nodes are decomposed and non-decomposed.

Node p is decomposed if it has at least one child p_i such that $\Psi_{pi} = \Psi_p$, a non-decomposed node can have at most one child p_i and the child satisfies $\Psi_{pi} \neq \Psi_p$.

Condition states that each node p in CAT tree is either a decomposed node in a conjunctive and disjunctive (AND, OR) manner.

$T(p) \in \{OR^A, AND^A, OR^C, AND^C\}$. Or is a non-decomposed node

$T(p) \in \{\mathrm{B}^A \bigcup \mathrm{B}^C\}$

Definition 5-4

If there exist $i \in \mathrm{N}/\{0\}$ such that $p_i \in A$ and $\Psi_{pi} = \Psi_p$ and $T(p) \in AND^A$ then

$P_{AND}^A = \prod_{k=1}^{n} P_k$ where P_{AND}^A is the probability for nodes to propagate up toward the root with AND decomposed.

$C_{AND}^A = Min(C) - \sum_{k=1}^{n-1} C_k / 100$ where C_{AND}^A is the cost for nodes to propagate up toward the root with AND decomposed.

Definition 5-5

If there exist $i \in \mathrm{N}/\{0\}$ such that $p_i \in A$ and $\Psi_{pi} = \Psi_p$ and $T(p) \in OR^A$ then

$P_{OR}^A = 1 - (\prod_{k=1}^{n} (1 - P_k))$ where P_{OR}^A is the probability for nodes to propagate up toward the root with OR gate.

$C_{OR}^A = Max(C) - ((\sum_{k=1}^{n-1} p_k * C_k) / (\sum_{k=1}^{n} p_k / 100))$ where C_{OR}^A is the cost for nodes to propagate up toward the root with OR gate.

Definition 5-6

If there exist $i \in \mathrm{N}/\{0\}$ such that $p_i \in C$ and $\Psi_{pi} = \Psi_p$ and $T(p) \in OR^c$ then

$P_{OR}^C = \prod_{k=1}^{n} P_k$ where P_{OR}^C is the probability for nodes to propagate up toward the root with OR gate.

$C_{OR}^C = Min(C_k) - \sum_{k=1}^{n-1} C_k /100$ where C_{OR}^C is the cost for nodes to propagate up toward the root with OR gate.

Definition 5-7

If there exists $i \in \mathrm{N}/\{0\}$ such that $p_i \in C$ and $\Psi_{pi} = \Psi_p$ and $T(p) \in AND^c$ then

$P_{AND}^C = 1 - (\prod_{k=1}^{n} (1 - P_k)$ where P_{AND}^C is the probability for nodes to propagate up toward the root with AND gate.

$C_{AND}^C = Max(C) - ((\sum_{k=1}^{n-1} p_k * C_k)/(\sum_{k=1}^{n} p_k /100)$ where C_{AND}^C is the cost for nodes to propagate up toward the root with AND gate.

6.5 COUNTERMEASURES COST

A dynamic severity cost measures for each state using the Dynamic Vulnerability Scoring System (DVSS) based on Intrinsic, time-based and dynamic-cost metrics by combining related sub-scores to produce a unique severity of impact cost by modelling the problem's parameters in a mathematical framework. To adjust the countermeasure Cost value using a developed risk metrics, considering both the likelihood and dynamic impact for each vulnerability, the mitigation cost is introduced, to calculate the mitigation (countermeasure) cost (CMit). The analyst needs to assess the outcome of the mitigation of each attribute of likelihood and

impact, and then apply the same pattern to work out the values of the mitigation. The values have a negative real numbers (as the cost of an exploit in the DVSS and in CVSS is taking a positive real number, logically the mitigations should be in negative values), using the DVSS framework. In the classical countermeasure approaches, the cost value of mitigation will be just equal to the monetary value of the vulnerability, in other words:

$$CMit_V = Cost_v - CostM_v$$

The mitigations could be considered as a process to eliminate the impact of publishing vulnerabilities, always consider the mitigations Cost are mathematically equivalent to the vulnerabilities cost without giving enough consideration to the likelihood or to the new methods to exploit them, for that reason, the following assumption is applied

$$Cost_v \geq CostM_v$$

Example 5-1

To calculate the mitigation cost of the vulnerability "ICS-ALERT-12-166-01 Sielco Sistemi Winlog Buffer Overflow" that was published June 14, 2012.

Taken from http://www.tenable.com

A DVSS of the vulnerability metrics are:

Intrinsic vector parameters =9.3:

AV:	N	AC:	M	Au:	N	C:	C	I:	C	A:	C

Time-based vector parameters are(Temporal score= 9.3):

E:	H	RL:	U	RC:	C

$Cost_v$ vector parameters are:

TD:	H	CDP:	MH	CR:	H	IR:	H	AR:	H

To calculate the effective cost using Intrinsic, Time-based and Dynamic-cost factors in the same approach introduced in Chapter-3 in this thesis:

$$I = 10.41*(1-(1-CI)*(1-InI)*(1-AI))$$

$$I = 10.0$$

$$E_x = 20*AV*AC*Au$$

$$E_x = 8.6$$

$$AT = round^1\ (\mathbf{BS_E} * \mathbf{E} * \mathbf{R}_L * \mathbf{R}_C)$$

$$AT = 9.3$$

$$Cost_v = (round^1(AT + (10-AT)*(CDP)*(TD))*10$$

$$Cost_v = 96.0$$

After implementing the Sielco Sistemi update to repair these vulnerabilities, the following update parameters are reached:

Intrinsic vector parameters are:

AV:	N	AC:	H	Au:	S	C:	N	I:	P	A:	N

Time-base vector parameters are:

E:	H	RL:	OF	RC:	C

$Cost_v$ vector parameters are:

TD:	M	CDP:	MH	CR:	H	IR:	L	AR:	L

Then the effective cost after implementing the fix can be calculated as follows:

$$I = 10.41*(1-(1-CI)*(1-InI)*(1-AI))$$

$$I = 6.4$$

$$E_x = 20*AV*AC*Au$$

$$E_x = 8.6$$

$$BS = (round^1((0.6*I + 0.4*E_x - 1.5)*Function(I))$$

$$BS = 6.8$$

$$AT = round^1\ (\mathbf{BS_E} * \mathbf{E} * \mathbf{R}_L * \mathbf{R}_C)$$

$$AT = 5.0$$

$$Cost_v = (round^1(AT + (10 - AT) * (CDP) * (TD)) * 10$$

$$Cost_v = 65.0$$

It can be seen that the cost drops down sharply after applying the mitigation, but the risk is still there and for that reason ICS-CERT asked asset holders to take extra security actions.

If the above Figures are implemented to the CAT, then the mitigation cost (CMit) will be:

$$CMit_V = Cost_v - CostM_v$$

$$CMit_V = 96.0 - 65.0$$

$$CMit_V = 31.0$$

Example 5-2

"Cross-site scripting attacks

Synopsis: The remote web server is prone to cross-site scripting attacks.

Description: The remote host is running a web server that fails to adequately sanitize request strings of malicious JavaScript. By leveraging this issue, an attacker may be able to cause arbitrary HTML and script code to be executed in a user's browser within the security context of the affected site. CVE : CVE-2002-1060, CVE-2003-1543, CVE-2005-2453, CVE-2006-1681 BID : 5305, 7344, 7353, 8037, 14473, 17408 Other

Server Specs: Windows Server 2008 Standard (32-bit)

Service Pack 1, references: OSVDB: 4989, OSVDB:18525, OSVDB:24469,OSVDB:42314."

Taken from, http://www.tenable.com

Intrinsic vector parameters are: CVSS Base=4.3 (Risk Factor: Medium)

AV:	N	AC:	M	Au:	N	C:	N	I:	N	A:	N

Time-based vector parameters are(Temporal score= 4.1):

E:	F	RL:	U	RC:	C

$Cost_v$ vector parameters are:

TD:	M	CDP:	MH	CR:	L	IR:	M	AR:	L

To calculate the effective cost using base, temporal and environmental factors:

$$I = 10.41*(1-(1-CI)*(1-InI)*(1-AI))$$

$$I = 2.90$$

$$E_x = 20*AV*AC*Au$$

$$E_x = 8.6$$

$$BS = (round^1((0.6*I + 0.4*E_x - 1.5)*Function(I))$$

$$BS = 6.8$$

$$AT = round^1\ (\mathbf{BS_E} * \mathbf{E} * \mathbf{R}_L * \mathbf{R}_C)$$

$$AT = 4.3$$

$$Cost_v = (round^1(AT + (10 - AT)*(CDP)*(TD))*10$$

$$Cost_v = 59.0$$

After implementing the fix, the parameters changed as follows:

Intrinsic Score = 3.9:

AV:	N	AC:	H	Au:	S	C:	N	I:	P	A:	N

Time-based vector parameters are:

E:	F	RL:	OF	RC:	C

$Cost_v$ vector parameters are:

AV:	N	AC:	M	Au:	N	C:	N	I:	P	A:	N

$$I = 10.41*(1-(1-CI)*(1-InI)*(1-AI))$$

$$I = 2.90$$

$$E_x = 20 * AV * AC * Au$$

$$E_x = 3.9$$

$$BS = (round^1((0.6 * I + 0.4 * E_x - 1.5) * Function(I))$$

$$BS = 2.1$$

$$AT = round^1\ (\mathbf{BS_E} * \mathbf{E} * \mathbf{R}_L * \mathbf{R}_C)$$

$$AT = 2.0$$

$$Cost_v = (round^1(AT + (10 - AT) * (CDP) * (TD)) * 10$$

$$Cost_v = 44.0$$

$$CMit_V = Cost_v - CostM_v$$

$$CMit_V = 59.0 - 44.0$$

$$CMit_V = 14.0$$

6.6 ALGORITHMS TO CONSTRUCT DYNAMIC ATTACK AND COUNTERMEASURES TREES

The DCAT can be described as a dynamic CCAT enhanced by adding a set of vulnerability mitigations to each leaf node in CCAT. To stand for effective manners of vulnerability mitigations in DCAT, the system administrator will calculate the effective cost of possible mitigations in an attack scenario using DCAT algorithms to concentrate the entire impact of a specific vulnerability. To review how to construct attack and countermeasures trees, using the formal representation and definitions presented above, a basic algorithm to construct CCAT and CAT, including the calculated metrics up to the root node, is set out in Figures 6-2 and 6-4.

A DCAT algorithm calculates metrics from leaves up to the root to compute the effective dynamic cost and probability.

ORprocedure $^{A}_{OR}$ (Psk, Csk) represents the procedure to calculate the cost and the probability. If the node type is OR and the node represented in attack type, then the cost and probability are calculated according to the following rules:

$$P_{OR}^{A} = 1 - (\prod_{k=1}^{n} (1 - P_k)$$

$$C_{OR}^{A} = Max(C) - ((\sum_{k=1}^{n-1} p_k * C_k) / (\sum_{k=1}^{n} p_k / 100)$$

while ANDprocedure $^{A}_{AND}$ (Psk, Csk) represents the procedure to calculate the cost and the probability. If the nodes are AND refinement with root according to the Edge et al formulae.

$$P_{AND}^{A} = \prod_{k=1}^{n} P_k$$

$$C_{AND}^{A} == Min(C) - \sum_{k=1}^{n-1} C_k / 100$$

If the node type is OR and the node represented in countermeasure type the ORprocedure $^{C}_{OR}$ (-Psk, -Csk) then calculate the cost and probability according to the following formulae:

$$P_{OR}^{C} = \prod_{k=1}^{n} P_k$$

$$C_{OR}^{C} = Min(C) - \sum_{k=1}^{n-1} C_k / 100$$

If the node type is AND and the node represented in countermeasure type, the ORprocedure $^{C}_{AND}$ (-Psk, -Csk) then calculate the cost and probability according to the following formulae:

$$P_{AND}^{C} = 1 - (\prod_{k=1}^{n} (1 - P_k)$$

$$C_{AND}^{C} = Max(C) - ((\sum_{k=1}^{n-1} p_k * C_k) / (\sum_{k=1}^{n} p_k / 100)$$

The DCAT approach enables the security decision making to calculate the expected loss before and after an investment decision, as exhibited in Section 5.5 in this chapter using attacks and countermeasures scenario.

Using different sets of countermeasures will result in different effective costs and probabilities that will help the network administrator to select the mitigation approach based on cost and benefits.

Using DCAT algorithm, help to regulate the risk and uncertainty in the enterprise architecture, to be reflected in term of cost-centric and mitigation cost attributes and calculate the influence of the mitigation attributes on the attacker's ability to compromise a vital asset.

A basic algorithm to construct CAT based on cost centric and considers attack and countermeasure costs, probabilities in one static tree structure, including the calculated metrics up to root node are shown in Figure 6-4,

Input: CAT elements with leaf nodes set L= {L_1, L_2,…, L_n}
Success attack probability s $\in S_n$, Success attack cost c $\in$ Cn. α = Node Type (A/C)
Output: The Cost and probability of the root node (Attacker goal)
Begin
Assign initial values (Probability and Cost) for all leaf nodes
For $\forall L_x \in \{L_1, L_2, \ldots, L_n\}$ **Do**
End For
Assign metrics for all leaf nodes
For x: =*1* to *n* Do
If the node.type="OR" * α = **"A"** *\

Psx, Csx $\leftarrow$ **ORprocedure (** P_{Sk}^A, C_{Sk}^A **)**

$$P_{Sk}^A := 1 - (\prod_{k=1}^{n}(1 - P_k)$$

$$C_{Sk}^A := Max(C_k) - ((\sum_{k=1}^{n-1} Ps_k * C_k)/(\sum_{k=1}^{n} Ps_k /100$$

Else

Psx, Csx $\leftarrow$ **ANDprocedure (** P_{Sk}^A, C_{Sk}^A **)**

$$P_{Sk}^A := \prod_{k=1}^{ns} P_k$$

$$C_{Sk}^A := Min(C_k) - \sum_{k=1}^{ns-1} C_k /100$$

End If
If the node.type="AND" * α = **"C"** *\

Psx, Csx $\leftarrow$ **ANDCprocedure (-** P_{Sk}^C, **-** C_{Sk}^C **)**

$$P_{Sk}^C := 1 - (\prod_{k=1}^{n}(1 - P_k)$$

$$C_{Sk}^C := Max(C_k) - ((\sum_{k=1}^{n-1} Ps_k * C_k)/(\sum_{k=1}^{n} Ps_k /100$$

Else

Psx, Csx $\leftarrow$ **ORCprocedure (-** P_{Sk}^C, **-** C_{Sk}^C **)**

$$P_{Sk}^C := \prod_{k=1}^{ns} P_k$$

$$C_{Sk}^C := Min(C_k) - \sum_{k=1}^{ns-1} C_k /100$$

End IF
End For
End

Figure 6-4. Algorithm of building Dynamic Countermeasures Attack Tree

6.7 EXAMPLE OF DCAT ANALYSIS

The Nessus scanner is employed to identify known vulnerabilities on PC1 and PC2 on a prototype network shown in Figure-3-5, presented in Chapter-3, for the evaluation of DCAT algorithms.

The following standard scheme is used to build the DCAT:

1. Using CostRank Algorithm to determine the critical asset the system administrator needs to protect;
2. Track the Nessus scanner to determine the vulnerabilities associated with the asset using the cost centric framework and DVSS;
3. Check out all published mitigations for all vulnerabilities.
4. Assess how well the security solution mitigates those risks;
5. Assess DCAT algorithm in terms of effective security cost and probability.

The foremost step is to select the critical assets that should be protected and choose PC1 from the paradigm network, represented in Figure 3-5.

In the second step to determine the vulnerabilities associated with the asset using the cost centric framework and DVSS; referring to chapter 3 the following vulnerabilities were discovered by Nessus scanner and classified by operational level 0 and 1 and 3 as shown in Figure-6-5 and Table-6-1.

	Vn1	Vn2	Vn5	Vn10			OLC	Value
Probability	0.31	0.25	0.20	0.24				
None privileges	57	47	38	44			CPC1,0	56.31
	Vn4	Vn6	Vn9	Vn11				
Probability	0.23	0.31	0.26	0.19				
User privileges	52	71	60	44			CPC1,1	70.47
	Vn3	Vn7	Vn8					
Probability	0.33	0.33	0.33					
Root/Administrator privilege	89	90	87				CPC1,2	89.12

Table 6-1 The Vulnerabilities OLC of PC1

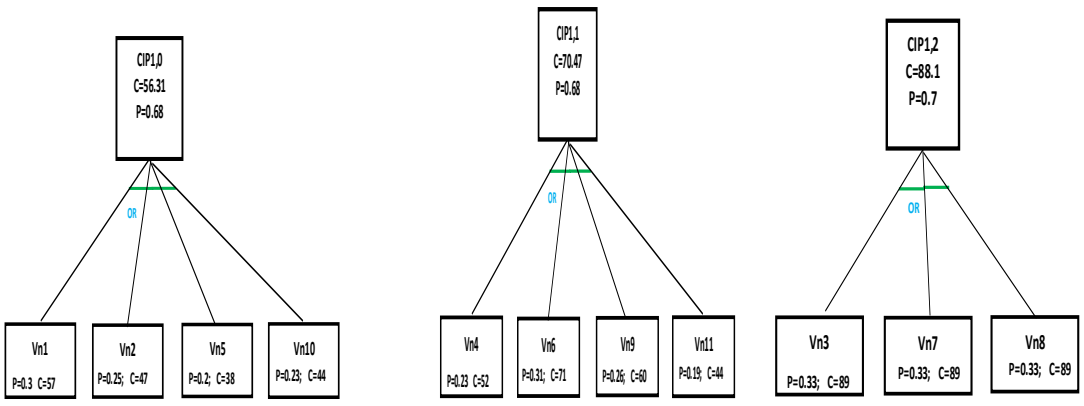


Figure 6-5. The vulnerabilities distributions of PC1

Figure-6-6 and Figure-6-7 show the vulnerabilities and mitigations in CCAT and DCAT are classified into three groups as user, root/administrator, and none. The user privileges mean that the intruder could exploit the vulnerabilities and gain a

user account privilege; while root privilege level refers to the vulnerabilities that could feed the intruder root or admin privilege after exploiting the vulnerabilities and none means the attackers can exploit the exposure from the networks or website with no privilege escalations. The following remark regarding the cost-centric value and probability transfer from the child nodes to the parent node:

1- The probability of parent node with AND bond will never increase as every attacker's action on the child nodes should succeed to find the parent node. The spread of the cost from the child node to parent's node measures as the norm for all child cost nodes, again will never increase.

2- The probability of parent node with OR bond never has dropping values because the attackers can reach the parents using different nodes, then the probability of the parent will be larger or equal to the chance of the minor nodes.

3- The cost-centric value of the parent node with OR bond will be near and less than the upper limit value of the child nodes and greater than the minimum value of the child nodes as one attacker's action on the child nodes should succeed to find the parent node. In other words, the spread of the cost from the child node to parent's node measures for all child cost nodes will never increase.

4- The cost-centric value of the parent node with AND bond will be near and less than the minimum value of the child nodes as all attacker actions on the child nodes should succeed to find the parent node. In other words, the spread of the cost from the child node to parent's node measures for all child cost nodes will never increase.

The majority of present CCAT and DCATs build on potential attack scenarios using attack profiles to detect possible vulnerable paths for the attackers to exploit vulnerabilities on the target assets. These approaches are in force in terms of finding vulnerabilities, but not in finding appropriate mitigations and measure the effective security before and later implementing the countermeasures. The DCAT

algorithm is developed based on dynamic cost centric, considered an attack and countermeasure costs, probabilities in one static tree structure.
The effectiveness of DCAT algorithm is measured in this case as the CAT algorithm is constructed based on vulnerability analysis using the Nessus scanner on critical and valuable asset computed using CostRank Algorithm, Dynamic Cost value calculated for each vulnerability and for each mitigation as shown in the examples in the first place.

As shown in Figure 6-7, possible mitigations for CCAT to protect PC1, represented in Figure 6-6, it can be ensured that the cost to make the root = 88.73 and the probability P= 0.95. That suggests a high risk as the root connected sub goal 1, 2, 3 using OR decomposed, as explained in the first place in this chapter, the solution for which is to implement the mitigations to sub goal-1, goal-2 and sub goal-3, as shown in Figure 6-7. The following mitigations will apply to the CCAT-Sub-Goal-2, shown in Figure 6-6:

Mv3-1, C= 67.4, P= 0.22.
Mv3-2, C= 79.4, P= 0.26.
Mv7-1, C=69.5, P=0.23.
Mv7-2, C= 79.8, P= 0.26.
Mv8, C= 79.5, P= 0.25.

There are two mitigations for Vulnerability-3 and Vulnerability-7, the Mv3-1 and Mv3-2, Mv7-1 and Mv7-2 with AND_K^C- According to the rules and definitions explained in this Chapter, the AND for countermeasures decomposed using $Max(C_k)$ and for that reason the CAT algorithm ignored the mitigation Mv3-1 and used Mv3-2 as Mv3-1 with higher effective cost weight and probability.

The following mitigations will apply to the CCAT- Sub-Goal-1:

Mv1, C=47, P= 0.27.

Mv2, C=43, P= 0.2.

Mv5. C=32, P= 0.1.

Mv10, C=38. P= 0.19.

The following mitigations will apply to the CCAT- Sub-Goal-3:

Mv9, C=50.8, P= 0.20.

Mv6, C=61.66, P=0.26.

Mv11, C=-41.66, P= 0.1.

Mv4, C=40.8, P= 0.2.

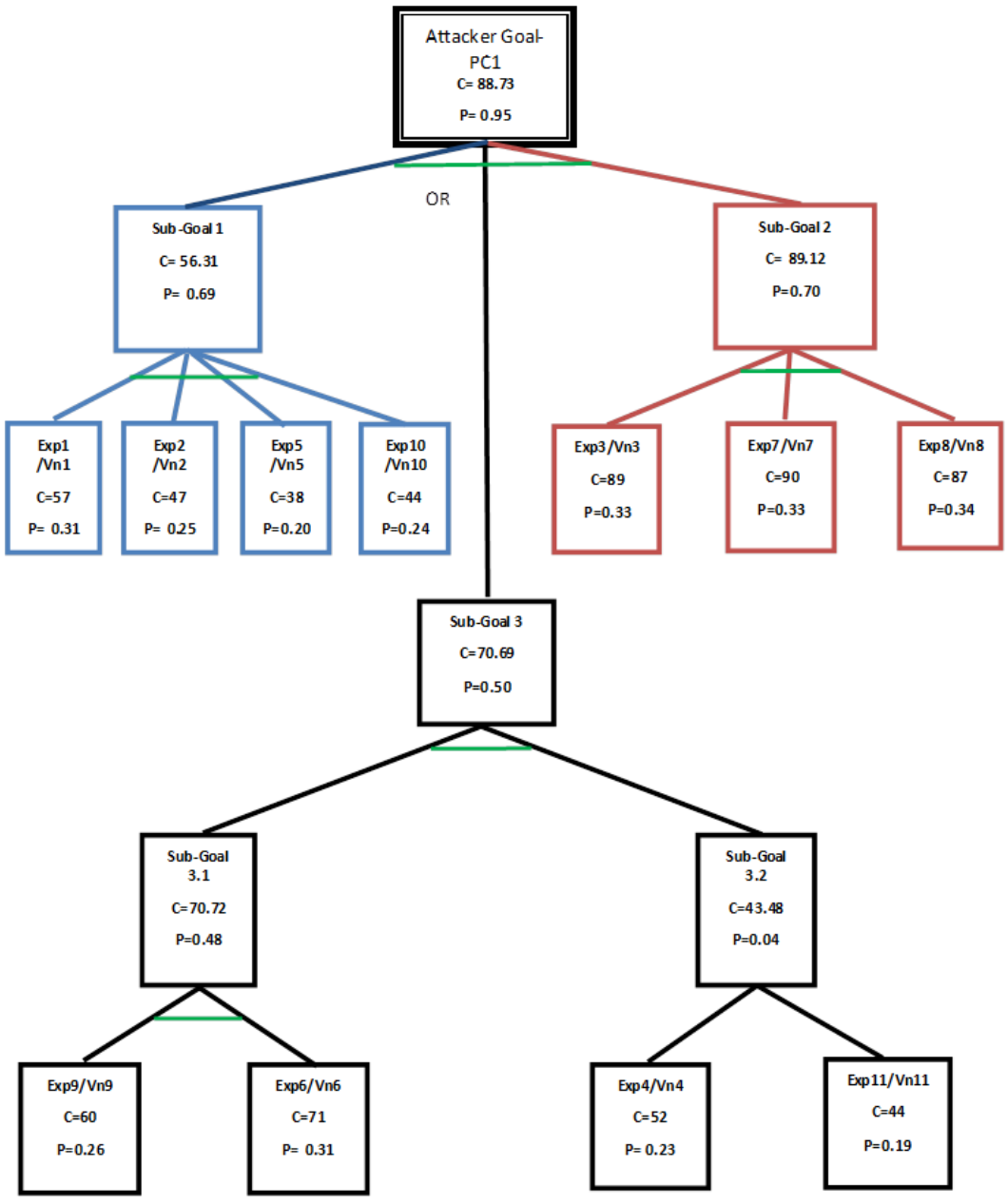


Figure 6-6. The DCCAT of PC1 uses OR decomposed for the root

The mitigations and security fixes in most cases are limited, as the software application providers need a long time to release the patches, since they require to carry out exhaustive testing to avoid more weaknesses. The most significance protection, when the network administrators experience problems in finding possible mitigations, is to use OR^C refinement arrangements to bypass the unpatched vulnerability and to reduce the probability that an attacker can reach the goal or move to the upper level. Otherwise, the mitigation selection will be according to the effective cost weight and probability. The weighting scale between the effective attack cost based on a “What - If” scenario and countermeasures cost represent the principal function of the CAT algorithm. The heightened DCAT is shown in Figure 6.7. As you will see, the cost value of exploiting PC1 drops down from 88.73 to 10.14, with an effective mitigations cost value of (-78.59), if the countermeasures and mitigations implemented to all vulnerabilities, according to cost/benefit principle, the system administrator could only implement solutions to some vulnerabilities depending on the importance of the assets. The cost-centric value of Sub Goal-2 drops down from 89.12 to 11.35, with an effective mitigations cost value of (-77.76). The cost value of Sub Goal-1 drops down from 56.31 to 9.95, with an effective mitigations cost value of (-46.36). Finally, Sub Goal-3 drops down from 70.69 to 9.16, with an effective mitigations cost value of (-61.53). Figure 6-8 and Figure 6-9, show the cost values and probabilities and their dropped down and mitigations for all vulnerabilities and mitigations used in the example. Please note that the best security cost is when the value approaches zero and the worst cost when the value approaches 100. On the other hand, the probability of exploiting PC1 drops down from 0.95 to 0.18.

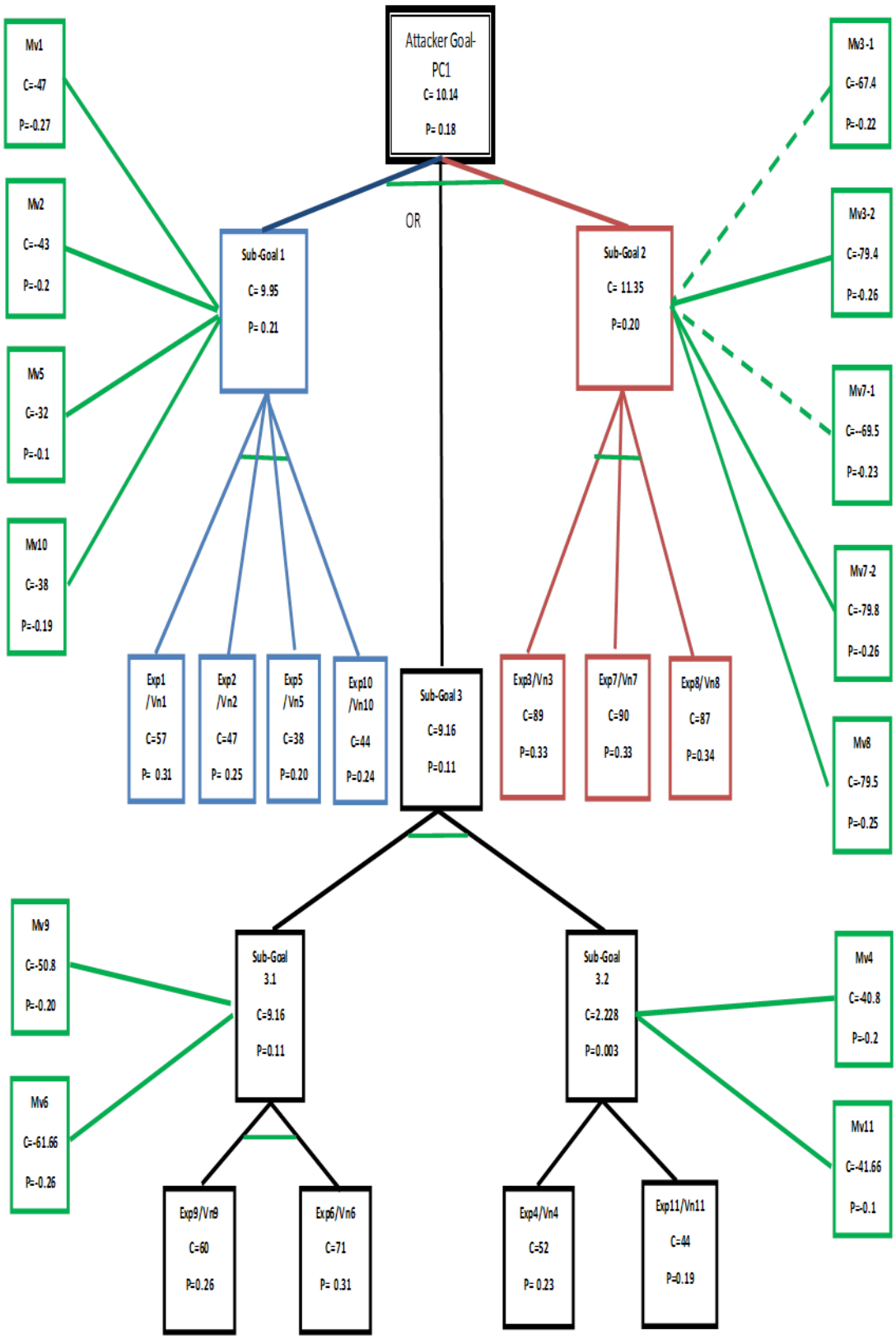


Figure 6-7.The optimized Dynamic Countermeasures Attack tree

The implementations of DCAT in the above example change the security view dramatically and offer better security in terms of cost and probability.

Figure 6-8, which represents a comparative figure of initial, mitigation, and effective cost value between CCAT and DCAT, shows clearly the improvement in security effective cost.

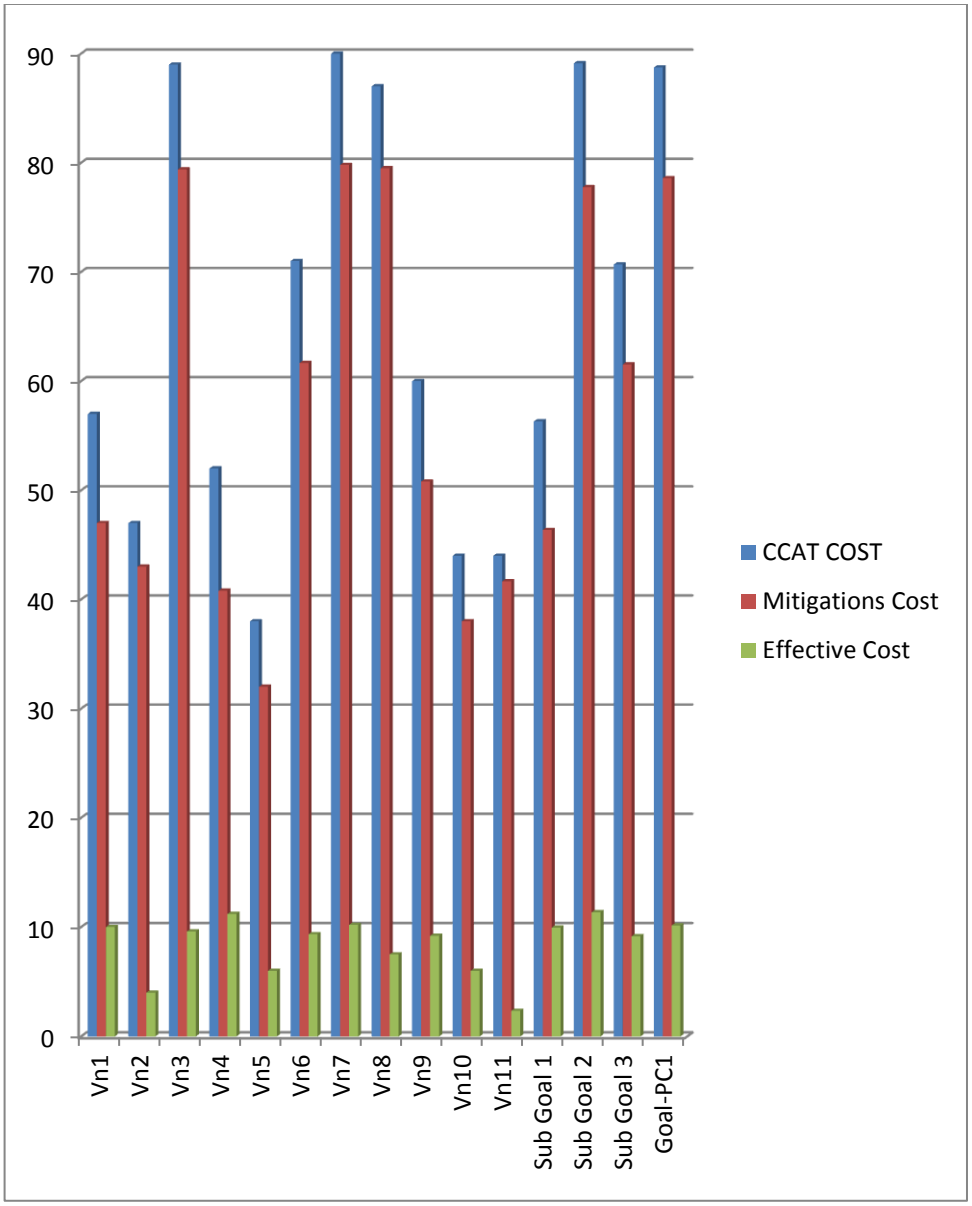


Figure 6-8. Comparative figure of vulnerabilities Cost between CCAT and DCAT

Figure 6- 9 demonstrates the comparative diagram of the probabilities of exploiting PC1; you can see the dramatic improvement in terms of reducing the probability value of PC1, the probability drops down from 0.95 to 0.18, with an effective mitigations probability value of (-0.77). The probability value of Sub Goal-1 drops down from 0.69 to 0.21, with an effective mitigations probability value of (-0.48). The probability value of Sub Goal-2 drops down from 0.7 to 0.2, with an effective mitigations probability value of (-0.5). Finally, Sub Goal-3 drops down from 0.5 to 0.11, with an effective mitigations probability value of (-0.39), before and after implementing CAT.

As you can see from the numerical analyses of DCAT algorithm, the effective countermeasures methodology developed against network attacks, applying different sets of countermeasures will result in different effective costs and probabilities, which help the network administrator to select the mitigation based on cost and benefits. The primary intention of developing DCATs is to combine countermeasures into directed acyclic graphs and to identify the critical threats in the systems to be mitigated. By selecting an effective, economical method to protect a critical asset by applying the mitigations only to a specific vulnerability, that bear on the critical assets and keeping off the total sum of mitigations to all exposures in the initiative.

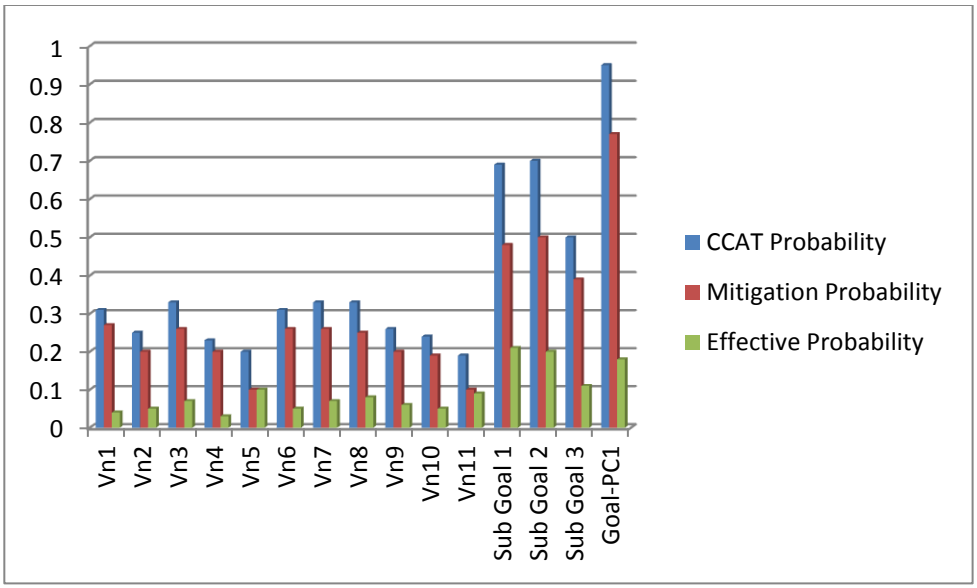


Figure 6-9.Comparative figure of probabilities between CCAT and CAT

6.8 BUSINESS CASE STUDY

In this section, a systematic study of the cost analysis of the dynamic cost-centric attack graph and mitigations is presented. In the area of security management, untreated threats can be hazardous and if the network administrators and security professionals are unaware of an attack, the impact could be very damaging to the businesses concerned. Referring to [Anderson, Ross, et al., 2012] model, the approximated costs of an attack are computed as the costs represent the expectancy of an approach, such as the prices of purchases of mitigations and implementing the countermeasures and controls by security pros. And, the consequence of an attack, such as the direct and collateral losses, and the response to an attach, such as recompense charges to the victims and other parties and Indirect costs, such as reputational impact and the loss as consequences of individual and business damage of internet transactions.

The business actions, customer satisfactions and overall national security depend on the security of network systems and cyberspace, including the software applications, data and the different technologies used to construct the network.

The Centre of Strategies and International Studies (CSIS) July 2013 estimated of criminal activity cost on the cyberspace in the range of $300 billion to $1 trillion and estimated cost of breaching the secrecy (privacy) in the range of $1 billion to $16 billion. As a result of the above Figures, billions of dollars have been allocated to secure the data of the public and private agencies, for instance, US government spends more than $100 million in the process of protecting, detecting and preventing cyber stacks.

In malice of all these expenses, the security still not in a good shape, As an example [Anderson, Ross, et al., 2012.] states, there are in the UK a little over $1,2bn estimated cost of the fraudulence of the DWP (Department of Work and Pensions), and $3bn Income tax fraud. Whereas, they are in the process of preceding all the claims online in 2013, which could make the number is a lot more eminent.

The attackers develop their skills and accomplished to discover new technique to exploit the vulnerabilities, even after implementing the mitigations or after obtaining a solution of a specific problem, using the multi - step approach. For example, the Google Chrome Pwnium browser attacks in 2012, as the attacker Pinkie Pie managed to perform a remote arbitrary code in the browser; using six individual low scoring vulnerabilities, in a successful multi-step approach to crack the Chrome sandbox.

The static approach of numerical scoring of vulnerabilities, ignoring the dynamic relations between vulnerabilities and ignoring the mitigations of some exposures with very small scoring impact, highlight the reasons of the current security problems, which lead to inappropriate policy decisions.

This thesis presented, a novel dynamic approach to calculate the severity of vulnerabilities, using a dynamic cost-centric approach. In order to formulate a cost-centric attack graph, using scalable parallel CostRank algorithms and finally using dynamic cost centric risk mitigation model are presented to improve the network and cyberspace security and help the security professionals and system administrators to support security decisions and to create and maintain security policies and procedures.

In this section, the approach of analysis employed in this business case study will discuss, to assist the security investment decision makers in the process of choosing a proper security solution based on the cost/benefit analysis.
Every bit you can understand from the numerical analyses of DCAT algorithm in Section 6-7, the effective dynamic countermeasures methodology developed against network attacks, using different sets of countermeasures will result in different effective costs and probabilities which help the network administrator to select the mitigation based on cost and benefit analysis. The primary intention of developing DCATs is to combine countermeasures into directed acyclic graphs and to identify the critical threats in the systems to be mitigated. By selecting an effective, economical method to protect a critical asset by applying the mitigations only to a specific vulnerability, that bear on the critical assets and keeping off the total sum of mitigations to all exposures in the initiative. This method will trim down the total price of implementing the countermeasures and ensure the effective protections of a critical asset.
The case study approach used in this thesis based on Daniel Bernoulli using Expected Utility Theory (EUT) and von Neumann-Morgenstern EUT [Mongin, 1997], these approaches calculate the consequence of the decisions under uncertainty and risk. As this thesis focuses on the risk valuation of a possible attack and determining a relevant solution of vulnerability mitigations to provide security professionals and system administrators with techniques and tools to support security decisions, the Bernoulli equations of probability theory and relate work is indispensable to assess the dynamic impact security, protection model. The annual loss, which interpret the likely loss per period (yearly in this case), can be calculated by the inbuilt of the possible loss probability with probability values of losses with a specific security level and without.
Let assume the loss measured with implementing a specific level of security $=L_{S1}$.
In addition, the loss measured with implementing another level of security=L_{S2}.
Then can calculate the annual lost as follows:

$$L_{S1-\text{Annual}} = \int_0^{\infty} x.L_{S1}(x)dx$$

$$L_{S2-\text{Annual}} = \int_0^{\infty} x.L_{S2}(x)dx$$

where x represents the possible loss.

To set the standard Bernoulli equations with probability theory, assume the probability of loss using the loss assumption with a specific security level=P_s, the probability of loss without implementing any security = (1-P_s) and α represents the expected impact, in this case the L_S can be calculated for assumption to have simple two loss outcome (0, α) as follows:

$$L_S = P_S.\alpha + (1 - P_S).0 = P_{S.}.\alpha$$

The expected benefit (β_S) of implementing a specific level of information security techniques could be calculated as follows Annual:

$$\beta_S = L_{S1-Annual} - L_{S2-Annual} = \int_0^{\infty} x.(L_{S1} - L_{S2})dx$$

$$\beta_S = \alpha.(P_{S1} - P_{S2})$$

The net benefit expected ($N_{\beta s}$) can be calculated as follows:

$N_{\beta s} = \beta_s - C_S$ where C_s= the cost of implementing a level of security.

Referencing to [Anderson, Ross, et al., 2012.] online banking attack cost:

Phishing: estimate= $16m, Global estimate= $320m.
Malware (consumer): UK estimate= $4m, Global estimate=$70m.
Malware (businesses): UK estimate= $6m, Global estimate=$300m.
The total estimated loss of indirect cost of £450 m ($700M).
Bank tech. Countermeasures: UK estimate= $50m, Global estimate=$1 000m.
In the dynamic cost-centric approach, to formulate a cost-centric attack graph, using scalable parallel CostRank algorithms and finally using dynamic cost centric risk mitigation model reduced the total cost of implementing the countermeasures and ensure the effective protections of a critical asset.

If the online banking attack in the UK is considered in this case study, considering the estimated price for implementing the countermeasures will reduce by only 7%, and the chance of successful attack will cut by only 3% using dynamic cost-centric approach.

The [Anderson, Ross, et al., 2012.] online banking attack estimate cost, change according above probability as follows:

Phishing: UK estimate= $25.5m, estimate= $319.9m.

Malware (consumer): UK estimate= $3.88m, Global estimate=$67. 9m

Malware (businesses): UK estimate= $5.82m, Global estimate=$291m

The total estimated loss of indirect cost of £436.5 m ($679M)

Bank tech. Countermeasures: UK estimate= $35m, Global estimate=$93 m

Now if the expected benefit is modified for 10 years project ($Time_{Max}=10$) with the yearly recruitment (c_R) cost of 1m in the dynamic security proposal and 5m yearly recruitment to mitigate all the vulnerabilities.

$$\beta_S = Time_{Max}.(L_N - L_S - c_R) - C_S$$

The expected benefit of 50m countermeasures is:

$$\beta_{St} = 10.(960 - 710 - 3) - 50 = 196m.$$

The expected benefit of the dynamic Security investment approach is:

$$\beta_{Sd} = 10.(960 - 688.7 - 1) - 35 = 235.3m.$$

With above calculations favour the dynamic approach, but let's use the net present value.

The return on security investment ($\Re$) is defined as the ratio of the benefit of the Security investment to the cost of security is:

$$\Re = \frac{N_S}{C_S} = \frac{\beta_S - C_S}{C_S} = \frac{L_{S1} - L_{S2} - C_S}{C_S}$$

$$\Re = \frac{L_{S1} - L_{S2} - C_S}{C_S} = -0.574$$

The net present value (P_n) considered the expected net benefit of security investment over the futures interval in the corresponding present value.

$$P_n = -C_{t=0} + \sum_{t=1}^{Time_{Mac}} \frac{L_{N,t} - L_{S,t} - c_r}{(1 - D_r)^t}$$

Let assume D_r=5% and $Time_{Max}$=10.

$P_{n-s} = 450m$ for the classical approach.

$P_{n-d} = 950m$ for the dynamic Security investment approach.

The internal rate of return (IRR) represents the discount rate (D_r) when P_n=0,

$D_r = 5.6$ for the classical approach.

$D_r = 9.3$ for the dynamic Security investment approach

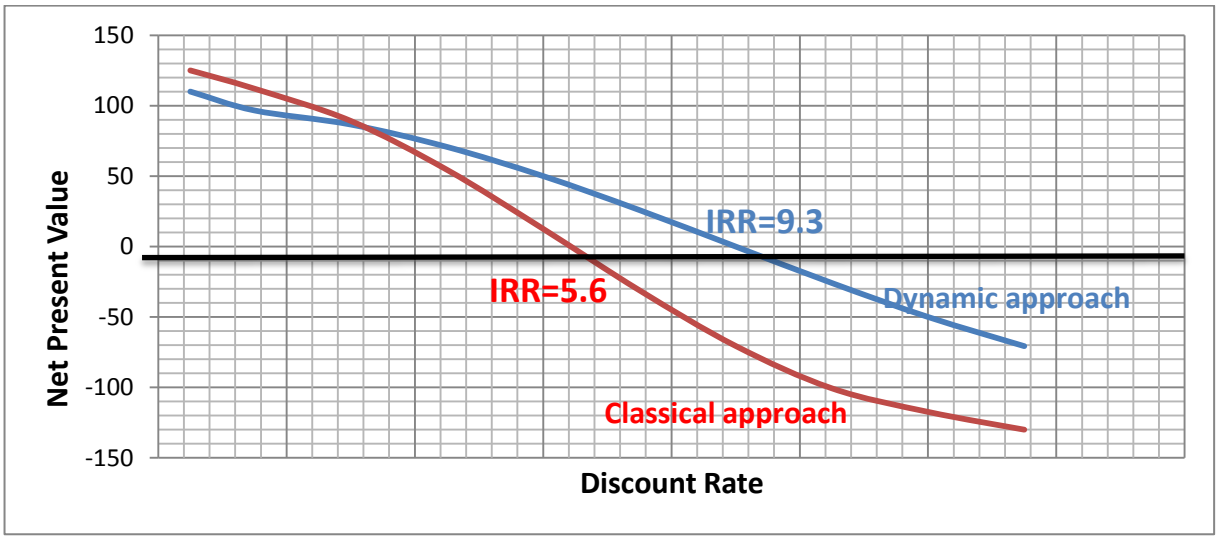


Figure 6-10. The visualisation of Net present value and IRR.

As you can see in Figure 6-10 higher rate of return means the security investment of the dynamic approach is more valuable.

In this segment, the business case study is presented, using security investment analysis, based on Daniel Bernoulli using Expected Utility Theory to forecast the expected benefits and the net expected benefit and Internal Net Return (IRR) of a dynamic security approach against the standard approaches of mitigations.

By using a humble figure of 7% reduction of the estimated cost, for implementing the dynamic countermeasures approach (as using dynamic cost centric risk

mitigation model reduced the total number of vulnerabilities mitigations, offer using a novel techniques to implement the mitigations and countermeasures for a critical vulnerabilities). The probability of successful attack is reduced by only 3% using dynamic cost-centric approach, which consider the interactions between vulnerabilities and finding a solution for treating a low score vulnerability and other parameters are covered in Chapter-3. The existing result of using two approaches is showing that the dynamic cost-centric approach provides better and cost effective solution compared with the classical methods of security protections.

6.9 SUMMARY OF FRAMEWORK DEVELOPMENT

In this chapter, a novel effective countermeasure-based solution is developed against network attacks using the cost-centric model checking by identifying the set of top network risks and hosts and application vulnerabilities, and countermeasures. In the DCAT model, attack detection and mitigation are permitted not just at the leaf node, but also at the transiting nodes as well as demonstrated in sub-goal-1, sub-goal-2 and sub-goal-3. In this example, the results of incorporating countermeasures and attacks into the DCAT, using a DVSS framework and Nessus scanner for vulnerability detections to construct both dynamic attack trees and DCAT are presented. The idea of mixing countermeasures into attack trees, and more generally into directed acyclic graphs using cost centric driven from DVSS and probability, is a new approach. The main difference between a dynamic CCAT and a DCAT is: that a CCAT is represents the attacker actions using dynamic cost centric values and probability. However, DCAT added a set of vulnerability mitigations to select an effective set of mitigations to reduce the possible damage by an attacker to a critical asset in terms of security protections, to improve the overall security of the system in the cost/benefit approach. The root of the DCAT is linked to a critical, valuable asset that the system administrators need to secure and give special consideration. The DCAT can be described as CCAT enhanced by adding a set of vulnerability mitigations to each child node, to support security decisions and to create and

maintain security policies and procedures. DCAT represent optimal manners of vulnerability mitigations in CAT, the system administrator will calculate the effective cost of possible mitigations in an attack scenario using DCAT algorithms to reduce the total impact of a specific vulnerability.

Mitigation for a single vulnerability might be able to stop one or more attacks, on the other hand, multiple mitigations for multiple vulnerabilities may be able to prevent or stop one malicious attack. The valued assets of the state space model are obtained according to the implementations of a dynamic CostRank algorithm, in the standard security mechanisms, vulnerabilities are acknowledged using a vulnerability scanner like Nessus. Vulnerabilities forming procedure and vulnerability analysis using DCAT are essential to systematically mitigate the vulnerabilities and find economical and effective possible solutions for system protections. Calculating the effective security measures in terms of countermeasure implementations and having alternative choices to apply different mitigation approaches in orders to reach an optimal security protection level. The consequences of incorporating countermeasures in the DCAT are studied using the case studies presented in chapter 3, with the DVSS calculator and Nessus scanner for vulnerability detections and both DCCAT and DCAT are constructed. The analysis of cost value and probability to reach a specific goal has performed and compared to existing results to evaluate the effectiveness of the methodology.

In this Chapter, a business case study is presented; the information systems represent an asset that should be secured to ensure that the business standard in the right directions, as any breach of the security could cause operational losses, damage in the confidentiality and integrity and availability and corporate reputation.

In this case study, a consideration of reduction of estimated cost by 7% only using the dynamic cost-centric countermeasures, 3% of successful attacks.

These figures are much lower than the reality, as the DCAT novel approaches are designed to implement the countermeasures only for a critical asset that the security professionals and system administrators need to secure, in additions offers the selection of more effective mitigations, which reduces the total cost.

Regarding successful attacks, the DCAT approach considers the interactions of vulnerabilities, network architectures, frequency of attacks, and many other parameters covered in Chapter-3.

The existing results shows, benchmarking with the classical approach of implementing the countermeasures that the DCAT approach provides momentum effective solutions to protect and secure information systems,

CHAPTER 7. CONCLUSIONS AND FURTHER WORK

7.1 CONCLUSIONS

Securing a large and complex network is a challenging task for security professionals and system administrators, as they need to trace and protect all the network paths that an attacker could use to exploit a specific vulnerability. Attack graphs (AG) are the most relevant method of illustrating the paths that an attacker can use to compromise vulnerabilities across the network. However, the existing AG models suffer from visual complications that deter humans from fully comprehending the information presented. An additional barrier is the huge amount of data required to accurately construct the AG.

This thesis presents novel approaches and methodologies to secure Network systems from intended attackers by providing security professionals and system administrators with techniques and tools to support security decisions and to create and maintain security policies and procedures. The approaches used in this thesis are based upon adapted AG models and associated set of algorithmic analysis tools.

The first hypothesis of this thesis was to evaluate the method of developing a unique severity cost of all vulnerabilities existing in a network host using dynamic risk impact metric methods. The purpose being to ensure that the new approach improved the risk scoring method and applied to produce cost-centric access to thin out the visual complexity of attack graph as presented and benchmarked in this dissertation.

A novel approach to developing tools for dynamic vulnerabilities quantitative scoring was presented, including network topological analysis to measure the impact of the network devices and configurations, and considering the correction factor to change the score of the intrinsic metric on the final scoring value of vulnerabilities. As the network topology and configuration change, the formula adapts the scoring values from the developed dynamic database file instead of

constant empirical static values used by CVSS, taking into considerations the interaction between vulnerabilities to improve the risk detections and mitigations. The dynamic impact scoring of vulnerabilities is used to advance the cost-centric method to assign a unique severity cost for each host by dividing the scores of the vulnerabilities to three main levels of privileges: i) none; ii) user and iii) root. The approach then classifies these levels into operational levels to identify and calculate the severity cost of multi-step vulnerabilities. These approaches and methods that provide an important step to achieve the aim of this thesis, to secure network systems and to assist the security investment decision makers in the process of selecting a proper security solution based on the cost/benefit analysis, as the cost-centric method will be used in building and ranking a novel cost-centric attack graph.

The cost of the total weight of all vulnerabilities for each host has been used to develop a unique dynamic severity, using DVSS scores of intrinsic, time-based and ecological metrics by combining related sub-scores and modelling the problem's parameters into a mathematical framework. In the test pad, Nessus Scanner 4.x is used to discover known vulnerabilities. Open source programs have been developed to process and combine different dynamic files to produce dynamic scores, the CVSS calculator demonstrated in the Appendix- is used to benchmark the DVSS scores with CVSS to monitor the impact of using a dynamic approach of calculation vulnerabilities scoring.

As the risk should be evaluated in terms of maximum impact on an adverse event, in the risk matrix used formulae are modified to collaborate the risk equations. The method of vulnerability classification is divided into three main levels, according to privileges (user, root/administrator, none) and is tested by comparing the most critical vulnerability scoring. To make sure that the calculated values reflect a real representation of the vulnerabilities existing in the hosts and the operational levels of vulnerabilities states for each privilege are rigor, significance, and it will lead to correct results, at the time when the dynamic cost-centric values used to build and rank the attack graph. Using the cost-centric framework to classify the vulnerabilities to none, user and root privileges, and then finding the

scores to OLC, the practical experiments proved that this classification really reflects the use of the risk matrices. The calculated values are used to build the cost-centric attack graph.
As established above, the author evidenced that a unique severity cost of all vulnerabilities existing in a network host with a new method to construct a cost-centric model checking approach, using novel dynamic risk impact metric methods improved the quantitative risk impact metric.

A new methodology has been developed to represent an attack graph with dynamic cost metrics and also a new methodology to calculate and represent the cost for each host. The problem's parameters have been modelled into a cost-centric framework to be used in the experiments.
A scalable, dynamic CostRank algorithm for ranking the AG is developed using a new approach to represent the states by implementing a dynamic cost-centric driven from statistical probability of dynamic impact scoring for the vulnerabilities. The ranks of the states represent the significance of the states in the DCCAT. Thus, these rankings represent a metric that can be used by security professionals and system administrators to make different security decisions to improve the network security based on cost/benefits as demonstrated in Chapter-6. Ranking an AG signifies a useful approach to solve the problem of AG visual complexity by providing a sight to the critical area of vulnerabilities to perform the analysis and to discover the effective method to implement the mitigations.
The CostRank algorithm is developed based on Google PageRank algorithm to demonstrate significance high rank states based on dynamic cost probability that the attackers can use in multi-step attack to exploit a vulnerability in the target critical asset.
In the CostRank algorithm the information about pre-condition probability, not necessary to be known before hand for all states as the values can be derived from partial information to perform the ranking, however knowing the probability can produce more accurate results.

The CostRank algorithm exercises an iterative mathematical process to calculate the maximal eigenvector of a practiced matrix with hosts' dynamic cost values derived from designated file structures. A framework is implemented on a test network, using the Nessus scanner to discover known vulnerabilities, implement these results and to build and represent the dynamic cost centric attack graph using ranking algorithms (in a similar fashion to Mehta et al. 2006 and Kijsanayothin, 2010). However, instead of using vulnerabilities for each host, a CostRank Markov Model has developed, thereby reducing the complexity in the attack graph and reducing the problem of visibility and significantly approved that the dynamic CostRank methodology can provide the optimum solution of ranking attack graphs towards minimising the dimension and complexity of attack graphs. A systematic model for parallel CostRank algorithm based on performance has been developed. The model is built on a number of intellectual performance factors such as load balancing, speed up, delay and efficiency.
As demonstrated above, a dynamic CostRank methodology provided an effective solution with ranking attack graphs using a cost-centric approach, minimising the dimension and complexity of attack graphs, please refer to the hypothesis-2.

A novel parallel algorithm is presented in this thesis to implement CostRank for distributed parallel computers using multiprocessors. To reduce the complexity of the serial CostRank algorithm and to make CostRank calculations more effective in terms of the cost/benefits approach. As the number of hosts in the network is increased, the serial CostRank algorithm needs to read the source CostRank values of different states and store it in a buffer, then read the parameters in the binary graph file header. This has three components, the entire number of states, and the entire number of links, and the upper limit out degree, in the attack graph. A state entry record saves the essential structural data for a state such as state ID, the out-degree, and then forward links (Link 1, Link 2, and Link v) along with corresponding cost for each link (C1, C2, and Cv) respectively. The parallel in-memory Piccolo algorithm is selected to extensively reduce access share state stored in memory and iterations, which include table partitioning (local access),

synchronization of distributed table, checkpoint/restore and load balance/ and task scheduling. In the same way, the large scales networks are secured that require fast and reliable computing to calculate the ranking of enormous graphs with thousands of states and millions or links. In this, a focus on a parallel CostRank computational architecture on a cluster of PCs networked for partitioning calculations and two masters, for synchronisations to assess the complexity and scalability of the algorithms.

In particular, a partitioning of input data, graph files, and ranking vectors with an appropriate load balancing technique has reduced the runtime and hence scalability of large scale parallel CostRank calculations. In this thesis, an application case study of parallel CostRank calculations is presented using one-dimensional sparse matrix partitioning on a modified research page at Stanford University. It describes the link structure of the stanford.edu-domain from a September 2002 collection and contains 281903 pages with about 2.3 million links, outcomes in a major reduction in communication overhead and in runtime. To study its efficiency, several tests were performed using out-vector files synchronised from the real networked data. The given results are quite encouraging.

The efficiency and speedup increased when the number of processors increased for a fixed problem size, the exceptional at case of using two processors as the overhead of partitions of graph link files, load balancing, synchronisations influence the efficiency, speed up.

During the investigations of the behaviour of CostRank algorithm, which provide a linear speedup as expected when the number of processors increased with fixed problem size, the observation that a slow system have better speedup for both single thread with delay and without delay.

An analysis of the performance and scalability of the parallel CostRank algorithm is presented in this thesis. The empirical approach to evaluating the CostRank algorithm’s performance demonstrates a major reduction in communication overhead and in runtime. Several tests were performed using out-vector files synchronised from real networked data.

A visible logical dialogue had also provided in terms of I/O and synchronization rate, and memory utilisation, speedup gain by using parallel CostRank over the Serial CostRank algorithm, load balancing, processor loads, and efficiency of parallel CostRank algorithms.

The isoefficiency Metric of Scalability was used to measure the scalability of CostRank algorithm. If the number of processors increases using fixed problem size, then the efficiency of the parallel scheme should increase accordingly, using similar fashions, if the problem size is fixed and at the same time the efficiency becomes stable, when the number of processors increased then, this will indicate bad utilisation of the processing recourses.

An experiment was performed to investigate the scalability of the Parallel CostRank algorithm. As expected according to isoefficiency approach, the problem size increased with the number of the processors to keep a fixed level of efficiency, which indicates good utilisation of the processing resources in term of machines and network specifications setting.

A partition-centred CostRank algorithm has been developed that can efficiently run on a parallel background and the scalability (isoefficiency function) of the parallel CostRank algorithm is $S(P_S) = \Theta(P_S^2 \log P_S)$.

From the above discussion, the parallel CostRank algorithm makes the rank calculations for an increasing number of hosts in the networks in a quick, effective and scalable manner; please refer to the hypothesis-3.

A novel effective countermeasure solution against network system attacks was established using a cost-centric attack graph by developing DCAT, in the DCAT model, attack detection and mitigation are legitimate not just at the leaf node, but at the transiting nodes. The results of incorporating countermeasures and attacks in the DCAT, using the DVSS framework and the Nessus scanner for vulnerability detections to construct both dynamic attack trees and DCAT, show clearly the effectiveness and the significance of CAT algorithm. These are, both in terms of reducing the total impact of a specific vulnerability and giving the security professional/system administrators the capability to select more mitigation that are effective. The idea of mixing countermeasures into attack trees, and more

generally into directed acyclic graphs using cost centric driven from DVSS and probability, is a new novel approach. The root of the DCAT is linked to a critical, valuable asset that the system administrators need to secure and give special consideration. The DCAT could be described as CCAT enhanced by adding a set of vulnerability mitigations to each child node, To represent optimal manners of vulnerability mitigations in CAT, the system administrator will calculate the effective cost of possible mitigations in an attack scenario using DCAT algorithms to reduce the total impact of a specific vulnerability.

Mitigation for a single vulnerability might be able to stop one attack or more, on the other hand multiple mitigations for multiple vulnerabilities may be able to prevent or stop one malicious attack. In other hand, both single and multiple mitigations may not prevent any attack, especially at the time of using static, classical policy of mitigations, in DCAT approach only effective mitigations in dynamic impact methods, which ensure the implementing operative mitigations in cost/benefit tactic.

The valued assets of the state space model are obtained according to the implementations of a dynamic CostRank algorithm, in the standard security mechanisms, vulnerabilities are acknowledged using a vulnerability scanner like Nessus scanner. Vulnerabilities forming procedure and vulnerability analysis using DCAT are essential to systematically mitigate the vulnerabilities and find economical and effective possible solutions for system protections. Calculating the effective security measures in terms of countermeasure implementations and having the alternative choices to apply different mitigations to reach optimal security protections.

The analysis of cost value and probability to reach a specific goal, was performed and compared to existing results in order to evaluate the effectiveness of the methodology.

A Business case study is presented, using security investment analysis based on Daniel Bernoulli using Expected Utility Theory. In order to calculate the expected benefit and the net expected benefit and Internal Net Return (IRR) of the dynamic security investment approach against the standard approaches of mitigations.

In this case study, a consideration of reduction of estimated cost by 7% only using the dynamic cost-centric countermeasures, 3% of successful attacks.
These figures are much lower than the reality, as the author, novel approach is designed to implement the countermeasures only for a critical asset that the security professionals and system administrators need to secure. In addition, this offers the selection of more effective mitigations, which reduces the total cost.
Regarding successful attacks, the approach presented in this thesis considers the interactions of vulnerabilities, network architectures, frequency of attacks and many other parameters of a dynamic cost impact approach.
The DCAT approach has produced dynamic countermeasure attack tree model that provides the cost effective solution to implement operative countermeasures and mitigations.
As established above, the author evidenced that Countermeasures Attack Tree algorithms can represent the optimum solution to implement effective countermeasures and mitigations..

7.2 FURTHER WORK

In this section, the set out a further study related to the methodologies and techniques employed in this thesis are presented.

7.2.1 Using Shannon information entropy to calculate uncertainties

The risk assessment of any security system has a high level of uncertainty because usually probability and statistics are used to evaluate the security of different cyber security systems. Using Shannon entropy to represent the uncertainty of information used to calculate systems risk and entropy weight method, since the weight of the object index is normally used and points to the significant components of the index. The risk of security systems should be evaluated in terms of different security layers and protection. The security evaluation algorithm must be written to normalise the protective matrix and

calculate the entropy and the entropy weight, then the weight and paths are used to evaluate and calculate the total risk in the system and give the system administrator clear guidance on the vulnerable security entities.

The basic security evaluation algorithm is written to normalise the protective matrix and calculate the entropy and the entropy weight, then the weight could be used to evaluate and calculate the total risk in the system and give the system administrator clear guidance on the vulnerable security entities.

Mat lab could be used to execute the following algorithms to calculate the entropy and the entropy weight:

1- Construct evaluation matrix: specify the different cost values assigned to Security entities in the form of numerical matrix r_{xy}.

2- Calculate the normalised process.

$$A_{xy=}(r_{xy})_{mn}$$

$$r_{xy} = \frac{a_{xy} - \min_{x}\{a_{xy}\}}{\max\{a_{xy}\} - \min\{a_{xy}\}}$$

3- Calculate the Entropy: if the index information is smaller, the index provides the information, content bigger, so the index of small entropy has important value.

$$H(T) = -k\sum_{i-1}^{n} Ci \Pr_i \ln\frac{1}{((1-\Pr_i)} \quad \text{Where } k = \frac{1}{\ln n}$$

4- Calculate the index entropy weight.

$$W_x = \frac{1-H(T)_x}{n-\sum_{x=1}^{n} H(T)_x}\ ,\ W_x \in [0,1], \text{and } \sum_{x=1}^{n} W_x = 1.$$

The majority of the works in this thesis, such as risk evaluation methodologies and Cost-Centric attack graphs and CAT, are based on probability and statistics and therefore the results will have enormous amounts of uncertainty, which might affect the accuracy of risk assessments. Using Shannon information entropy and

entropy weight is suggested to represent the average uncertainties and combined them with the results to improve the quality. That related to objective-1 improved the quantitative risk impact metric, objective-2 improve the cost-centric approach based on an attack graph model, the objective-3 dynamic CostRank methodology, objective-4 parallel CostRank algorithm and objective 5 Countermeasure Attack Tree algorithms.

7.2.2 Combine the DCAT and the Dynamic Cost Centric AG, model with the library of action

As further work, referring to [Kannan, Karthik, and Rahul Telang, 2005], it is possible to combine the Countermeasures Attack Tree and the cost centric AG model with the library of action to automate the mitigation similar to Net SPA. An input needs to build to the model builder, obtaining information about the vulnerabilities and mitigations, and to build an atomic framework of attack graph and DCAT for better and more accurate results, as this action related to the thesis aim and all objectives.

7.2.3 Feasibility for very large graphs

In this work, the author started by highlighting the restrictions of previous work in network hardening using attack graphs. In confident, both theoretically and practically approach, shows that the model is feasible only for relatively small and medium graphs.

For future work, implementing both parallel dynamic CostRank and DCAT Algorithms for a larger graph is suggested to obtain more findings to correlate the algorithms and make them ready for industrial applications.

The wall clock time performance metric when the Parallel CostRank algorithm used different platforms and network specifications, reduces convergence time, extensive memory demanding metrics are not counted as they are not in the scope this thesis and they are suited to be investigated in the future work.

GLOSSARY

A

Attack trees: are an abstract diagram exhibition showing in what manner a target, or critical asset, might be attacked (the attackers exploit the vulnerabilities). Attack trees have been cast off in a variety of IT applications.

Attack: A violation of system security policies that originates from an existing vulnerability, i.e., an intellectual action (attempt) of an attacker to exploit existing vulnerabilities to escape security services and intrude upon the security policy of a system. The attack type, as active or passive can occur, by an inside or by outside attackers, or through a multi - step attack.

Attacker: Attacker is individual who endeavours to exploit one vulnerability or more to accomplish an objective(s).

B

Backdoor: A backdoor is an instrument fitted after an attack to enable an attacker escalates the privileges and gain access to the other systems all over the place, to violate existing security policies.

Bots: The Bots are ideal Trojan Horses. With the ability to communicate with the boat creator, which enables the attackers to control the communication channels. Some Bots, have update structures. At the beginning Bots comes as a spam spread. When that is sorted, they converted as Distributed Denial of Service architects. Recently converted as keystroke loggers chasing for financial or software license information. 70 to 80 percent of all spam originates from bots..

Buffer Overflow: A buffer overflow (buffer overrun), is an abnormality of using a program or application, though writing data to a buffer, invades the buffer limit and overwrites adjacent memory.

Buffer overflows can be started by inputs that are intended to execute scripts. This can produce unpredictable of application behaviour, together with memory access errors, improper outcomes, a crash, or a compromise of system security. The buffer

overflow can produce by using software vulnerabilities, multi –step attacks and constructed executable codes.

C

CCATs: Acronym for “Cost-Centric Attack Trees.” are conceptual diagrams showing how an asset, or target, might be attacked using the cost metric of vulnerabilities to represent the tree.

CCATs: Acronym for “Cost-Centric Attack Trees.” are conceptual diagrams showing how an asset, or target, might be attacked using the cost metric of vulnerabilities to represent the tree.

CERT: Acronym for “Computer Emergency Response Team.”, the CERT is developed by Carnegie Mellon University to use of applicable technology to avert attackers from exploiting vulnerabilities in network systems, to reduce attack impacts and to guarantee the availability of significant services.

Compromise: a compromise is a term used in security to designate an occurrence of exposed confidential data to unofficial individuals. The release of the private information is expected to have a bad economic impact on the organisation's contour and legal, reputation effect. The compromise can be either intentional or unintentional.

Countermeasure: a countermeasure is an inaction, such mitigation policy, procedure, or technique that decreases the harm (impact) of a vulnerability, preventing vulnerability exploits (an attack) or by detecting and reporting the vulnerability, consequently a remedial action can be taken.

CostRank algorithm: Is a ranking algorithm that calculates the costs of a dependency AG state using a cost-centric method. The algorithm analyses both the number of incoming links and the cost of a specific state based on cost-centric framework of the referring vulnerability to generate an effective cost measurement.

CVSS: Acronym for “Common Vulnerability Scoring System.” An industry open standard, CVSS is comprised of three metric sets: Base, Temporal and Environmental metrics; respectively containing established numerical severity

quantitative measures aimed to provide the end user with a complete combination mark to represent the static risk and severity of a vulnerability.

D

Damage: The damage represents the consequences of an attack (intended or unintended), which might affect the confidentiality, integrity and the availability of the processes of the target system.

Data: Detailed annotations, usually in numeric or textual form.

DCAT: Acronym for “Dynamic Countermeasures Attack Tree.” are conceptual diagrams showing how an asset, or target, might be attacked using the dynamic cost metric of vulnerabilities and the dynamic cost metric of mitigations to calculate the effective cost of implementing the mitigations.

Defence Trees: a type of ATs has been developed to investigate the effect of defence mechanisms using measures such as attack cost, security investment cost’s weight, return on attack (ROA), and return on investment (ROI).

139.

Denial of Service: The process of preventing authorised users from accessing the system resource or causing a delay in accessing the system operations and services.

DVSS: Acronym for “Dynamic Vulnerability Scoring System.” An improvement of CVSS, is comprised of three metric sets: Intrinsic, Time-based and Ecological metrics; respectively containing established numerical dynamic severity quantitative measures aimed to provide the end user with a complete combination mark to represent the dynamic risk and severity of a vulnerability.

Dynamic systems: Qualitative observational research is not apprehensive with having direct, right, or wrong responses. Alteration in a study is a normal process as the researcher's aim is not product only one solution.

E

Efficiency: This term is generally used in the parallel algorithms literature to mean the speedup divided by the number of units of execution such as the number of processors.

F

Firewall: Is a software/hardware tools, designed to secure a network resources from unauthorised users, by denying/permitting a specific traffic between networks, utilising different security levels using sets of rules and policies.

H

Hijack Attack: is the types of tapping, as the attackers take control on established communication channels, violate the confidentiality and integrity and availability.

Host: Is a computing Hardware with processors as the operating system and software applications are based.

I

Impact: Impact represents the results (damage) caused by an attack in term of cost such as CIA disruption, financial consequences.

Injection Flaws: this type of attack can occur when the attackers sends (inject) untrustworthy text/data to, for example, LDAP, SQL queries or OS injection, the injection source data will have a code to exploit the syntax vulnerability of the directed interpreter and might result in account takeover, retrieving data/information, data loss/corruptions or denial of access.

IP Address: denoted as “internet protocol address.” Is a numerical code that identifies a specific computing device on the Internet.

L

LAN: Acronym for “local area network.” Is a cluster of computing devices connected through infrastructure backbone to share resources and common communications.

Link: is a pointer from one vertex (node) to another.

M

Malware: A common term for "malicious software". Represent any software (such as programs, Files, emails) that cause a harm to the computing devices, examples of malware are viruses, Trojon horses, worms and spyware.

Markov Chain: A Markov Chain is a stochastic model recounting a sequence of possible actions in which the probability of each action is subject to only the state achieved in the previous action.

Mitigation: Is a methodical decrease in the cost of a vulnerability/risk in term of likelihood and impact. Sometime it called “risk reduction”.

N

NIST: Acronym for “National Institute of Standards and Technology.” Non-regulatory federal agency within U.S. Commerce Department's Technology Administration.

NVD: Acronym for “National Vulnerability Database.” Is one of the National Institute of Standards and Technology (NIST)’s important security assets for determining the severity of computer security risks. NVD is the sum of many other security databases and utilizes the CVSS scoring system.

P

Patch: is an update to existing applications to enhance functionality or to fix a vulnerability or problems.

Policy: represent rules of principal satisfactory use of computing resources, security put into practise, and managerial development of operative procedures

Probability: Is the fortuitous that an occurrence is going on randomly, the degree to which something is probable to occur or be the instance.

Processor: is the logical electrical system that reacts to and processes the different instructions that tell the computer what to do.

R

Rigour: Grade to which research methods are carefully and accurately carried out in order to distinguish significant impacts happening in a experimentation.

Rootkit: A set of software tools and techniques that facilitate an attacker to escalate the privilege to root/administrator level, in order to control a computer system.

S

Speedup: Is the ratio of decline in the run time required to process a specific problem size (a task) using many processors (Parallel) to the run time required to process same task using a single processor (Serial).

SQL Injection: SQL injection is a security exploit, where the attacker develops a crafted Structured Query Language (SQL) script in the input box of a specific Web site to gain access, modify the resources, influencing the integrity and confidentiality of the system.

T

Trojan Horse: Is a useful, (generally free) computer software application/program having a significant malicious function to avoid a security restriction and performs a harmful task such as escalate the privileges to access system resources.

V

Visual complexity: visual complexity of an Attack graph, described the complexity, accessibility and usability problems of an AG, which deters humans from fully comprehend the information presented by the AG tools.

Vulnerability: Is a weakness/flaw in the design of software applications, configurations, operations and management, which can be exploited by attackers to compromise the CIA of system resources and evade the security policies.

WORKS CITED

Ammann, Paul, Duminda Wijesekera, and Saket Kaushik. "Scalable, graph-based network vulnerability analysis." Proceedings of the 9th ACM Conference on Computer and Communications Security. ACM, 2002.

Anderson, Ross, Chris Barton, Rainer Böhme, Richard Clayton, Michael van Eeten, Michael Levi, Tyler Moore, and Stefan Savage. "Measuring the Cost of Cybercrime." In WEIS. 2012.

Arasu, Arvind, et al. "PageRank computation and the structure of the web: Experiments and algorithms." Proceedings of the Eleventh International World Wide Web Conference, Poster Track. 2002.

Artz, Michael Lyle. Netspa: A network security planning architecture. Diss. Massachusetts Institute of Technology, 2002.

Aykanat, Cevdet, B. Barla Cambazoglu, and Bora Uçar. "Multi-level direct k-way hypergraph partitioning with multiple constraints and fixed vertices." Journal of Parallel and Distributed Computing 68.5 (2008): 609-625.

Bhattacharya, Somak, and S. K. Ghosh. "An attack graph based risk management approach of an enterprise lan." Journal of Information Assurance and Security (JIAS) 3.2 (2008).

Bilar, Daniel, and George Chairperson-Cybenko. "Quantitative risk analysis of computer networks." (2003).

Bistarelli, Stefano, Fabio Fioravanti, and Pamela Peretti. "Defense trees for economic evaluation of security investments." Availability, Reliability and Security, 2006. ARES 2006. The First International Conference on. IEEE, 2006.

Borodin, Allan, et al. "Link analysis ranking: algorithms, theory, and experiments." ACM Transactions on Internet Technology (TOIT) 5.1 (2005): 231-297.

Chen, Yen-Yu, Qingqing Gan, and Torsten Suel. "I/O-efficient techniques for computing PageRank." Proceedings of the eleventh international conference on Information and knowledge management. ACM, 2002.

Deraison, Renaud. "Nessus Security Scanner. The" Nessus" Project. Nessus Org. Nessus Homepage." (2006).

Ding, Chris, et al. "PageRank, HITS and a unified framework for link analysis." Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, 2002.

Edge, Kenneth S. A framework for analyzing and mitigating the vulnerabilities of complex systems via attack and protection trees. No. AFIT/DS/ENG/07-13. AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING AND MANAGEMENT, 2007.

F. Chen, A. Liu, Y. Zhang, J. Su, "A Scalable Approach to Analyzing Network Security using Compact Attack Graph," JOURNAL OF NETWORKS, VOL. 5, NO. 5, 2010.

Franqueira, Virginia NL, and Maurice van Keulen. "Analysis of the NIST database towards the composition of vulnerabilities in attack scenarios." Centre for Telematics and Information Technology (CTIT), University of Twente, Enschede, The Netherlands, Tech. Rep. TR-CTIT-08-08 (2008).

Frigault, Marcel, et al. "Measuring network security using dynamic bayesian network." Proceedings of the 4th ACM workshop on Quality of protection. ACM, 2008.

G. Helmer, Guy, et al. "A software fault tree approach to requirements analysis of an intrusion detection system." Requirements Engineering 7.4 (2002): 207-220.

Gene M. Amdahl. "Validity of the single processor approach to achieving large scale computing capabilities". AFIPS conference proceedings, April 1967.

Grama, Ananth Y., Anshul Gupta, and Vipin Kumar. "Isoefficiency: Measuring the scalability of parallel algorithms and architectures." IEEE concurrency 1.3 (1993): 12-21.

Granadillo, Gustavo Daniel Gonzalez. Optimization of cost-based threat response for Security Information and Event Management (SIEM) systems. Diss. Institut National des Télécommunications, 2013.

Gürdag, Adnan Burak. A PARALLEL IMPLEMENTATION OF A LINK–BASED RANKING ALGORITHM FOR WEB SEARCH ENGINES. Diss. Bogaziçi University, 2002.

He, Liwen, and Nikolai Bode. "Network Penetration Testing." EC2ND 2005. Springer London, 2006. 3-12. **Hendrickson, Bruce, and Tamara G. Kolda.** "Graph partitioning models for parallel computing." Parallel computing 26.12 (2000): 1519-1534.

Herrmann, Debra S. Complete guide to security and privacy metrics: measuring regulatory compliance, operational resilience, and ROI. CRC Press, 2007.

Houmb, Siv Hilde, and Virginia NL Franqueira. "Estimating ToE risk level using CVSS." Availability, Reliability and Security, 2009. ARES'09. International Conference on. IEEE, 2009.

Idika, Nwokedi C. CHARACTERIZING AND AGGREGATING ATTACK GRAPH-BASED. Diss. Purdue University, 2010.

J. Cullum, C. E. Irvine, and T. E. Levin. Performance impact of connectivity Restrictions and increased vulnerability presence on automated attack graph generation. In ICIW 2007: 2nd Int. Conf. on i-Warfare and Security Naval Postgraduate School, pages 33-46, England, March 2007. Academic Conferences Limited.

Jajodia, Sushil, Steven Noel, and Brian O'Berry. "Topological analysis of network attack vulnerability." Managing Cyber Threats. Springer US, 2005. 247-266.

Jürgenson, Aivo. Efficient Semantics of Parallel and Serial Models of Attack Trees. TUT Press, 2010.

K. Avrachenkov, N. Litvak, D. Nemirovsky, and N. Osipova. Monte Carlo methods in PageRank computation: When one iteration is sufficient. SIAM J. Numer. Anal., 45:890–904, 2007.

Kannan, Karthik, and Rahul Telang. "Market for software vulnerabilities? Think again." Management Science 51.5 (2005): 726-740.

Kizza, Joseph Migga. "Security Assessment, Analysis, and Assurance." Guide to Computer Network Security. Springer London, 2013. 145-168.

Kijsanayothin, Phongphun. Network security modeling with intelligent and complexity analysis. Diss. Texas Tech University, 2010.

Lattanzi, Silvio, et al. "Filtering: a method for solving graph problems in mapreduce." Proceedings of the twenty-third annual ACM symposium on Parallelism in algorithms and architectures. ACM, 2011.

L. G. Valiant, "A bridging model for parallel computation," Commun. ACM, vol. 33, no. 8, pp. 103–111, 1990.

L. Wang, A. Singhal, S. Jajodia, “Measuring overall security of network configurations using attack graphs,” Data and Applications Security XXI, vol. 4602, pp. 98–112, August 2007.

L. Wang, T. Islam, T. Long, A. Singhal, S. Jajodia, “An attack graph-based probabilistic security metric,” DAS 2008, LNCS 5094, pp. 283–296, 2008.

Li, Zhi-tang, et al. "A data mining approach to generating network attack graph for intrusion prediction." Fuzzy Systems and Knowledge Discovery, Fourth International Conference on. Vol. 4. IEEE, 2007.

LINDE, R. Operating System Penetration. In Proceedings of the National Computer Conference, Vol 44. AFIPS Press, Montvale, NJ, 1975.

Luo, Jian, Kueiming Lo, and Haoran Qu. "A Software Vulnerability Rating Approach Based on the Vulnerability Database." Journal of Applied Mathematics 2014 (2014).

M. Dacier. Towards Quantitative Evaluation of Computer Security. PhD thesis, Institut National Polytechnique de Toulouse, December 1994.

M.S. Ahmed, E. Al-Shaer, E. Khan, “A novel quantitative approach for measuring network security,” Proceedings of IEEE INFO COM 2008. Management, pages 578–585, 2005.

Manaskasemsak, Bundit, and Arnon Rungsawang. "An efficient partition-based parallel PageRank algorithm." Parallel and Distributed Systems, 2005. Proceedings. 11th International Conference on. Vol. 1. IEEE, 2005.

Massacci, Fabio, Stephan Neuhaus, and Viet Hung Nguyen. "After-life vulnerabilities: a study on firefox evolution, its vulnerabilities, and fixes." Engineering Secure Software and Systems. Springer Berlin Heidelberg, 2011. 195-208.

Mauw, Sjouke, and Martijn Oostdijk. "Foundations of attack trees." Information Security and Cryptology-ICISC 2005. Springer Berlin Heidelberg, 2006. 186-198.

Mehta, Vaibhav, et al. "Ranking attack graphs." Recent advances in intrusion detection. Springer Berlin Heidelberg, 2006.

Mell, Peter, Karen Scarfone, and Sasha Romanosky. "A complete guide to the common vulnerability scoring system version 2.0." Published by FIRST-Forum of Incident Response and Security Teams. 2007.

Mellado, Daniel, Eduardo Fernández-Medina, and Mario Piattini. "A comparative study of proposals for establishing security requirements for the development of secure information

systems." Computational Science and Its Applications-ICCSA 2006. Springer Berlin Heidelberg, 2006. 1044-1053.

Mongin, Philippe. "Expected utility theory." Handbook of economic methodology (1997): 342-350.

N.C. Idika, "Characterizing and Aggregating Attack Graph-Based Security Metrics," PhD Dissertation, Purdue University, West Lafayette, Indiana, 2010.

Noel, Steven, et al. "Efficient minimum-cost network hardening via exploit dependency graphs." Computer Security Applications Conference, 2003. Proceedings. 19th Annual. IEEE, 2003.

O' Hare, Scott, Steven Noel, and Kenneth Prole. "A graph-theoretic visualization approach to network risk analysis." Visualization for Computer Security. Springer Berlin Heidelberg, 2008. 60-67.

Opel, Alexander. "Design and implementation of a support tool for attack trees." Internship Thesis, Otto-von-Guericke University Magdeburg (March 2005) (2005).

Ou, Xinming, Sudhakar Govindavajhala, and Andrew W. Appel. "MulVAL: A logic-based network security analyzer." 14th USENIX Security Symposium. 2005.

Ou, Xinming, Wayne F. Boyer, and Miles A. McQueen. "A scalable approach to attack graph generation." Proceedings of the 13th ACM conference on Computer and communications security. ACM, 2006.

Oxford Dictionaries, http://oxforddictionaries.com/definition/english/metric. Accessed on 09/10/2013/

Papadaki, Maria, and Steven Furnell. "Vulnerability management: an attitude of mind?." Network Security 2010.10 (2010): 4-8.

Paul Ammann, Duminda Wijesekera, and Saket Kaushik. Scalable, graph-based network vulnerability analysis. In 9th ACM Conference on Computer and Communications Security, pages 217–224, 2002.

Power, Russell, and Jinyang Li. "Piccolo: Building Fast, Distributed Programs with Partitioned Tables." OSDI. 2010.

Prasad, Major Santosh. Development of Exploitation Framework for Vulnerability Assessment of Enterprise LAN. Diss. Indian Institute of Technology, 2008.

Purboyo, Tito Waluyo, and Budi Rahardjo. "Some framework, Architecture and Approach for analysis a network vulnerability." System Engineering and Technology (ICSET), 2012 International Conference on. IEEE, 2012.

R. Lippmann, K. Ingols, C. Scott, Piwowarski, K. Kratkiewicz, M. Artz, R. Cunningham, "Validating and restoring defence in depth using attack graphs," Military Communications Conference, October 2006.

R.W. Ritchey and P. Ammann. Using model checking to analyze network vulnerabilities. In Proceedings of the IEEE Symposium on Security and Privacy, pages 156–165, May 2001.

Roy, Arpan, Dong Seong Kim, and Kishor S. Trivedi. "Attack countermeasure trees (ACT): towards unifying the constructs of attack and defense trees." Security and Communication Networks 5.8 (2012): 929-943., PhD thesis 2010.

Rungsawang, Arnon, and Bundit Manaskasemsak. "PageRank computation using PC cluster." Recent Advances in Parallel Virtual Machine and Message Passing Interface. Springer Berlin Heidelberg, 2003. 152-159.

S. Kamvar, T. Haveliwala, and G. Golub. Adaptive methods for the computation of PageRank. Linear Algebra Appl., 386:51–65, 2004**.**

Sawilla, R., and X. M. Ou. Googling attack graphs. Defence R & D, Canada. Ottawa, Tech. Rep: TM 2007-205, 2007.

Sawilla, Reginald E., and Xinming Ou. Identifying critical attack assets in dependency attack graphs. Springer Berlin Heidelberg, 2008.

Sawilla, Reginald Elias. "Ranks and Partial Cuts in Forward Hypergraphs." (2011).

Schneier, Bruce, and Adam Shostack. "Breaking up is hard to do: modeling security threats for smart cards." USENIX Workshop on Smart Card Technology, Chicago, Illinois, USA, http://www. counterpane. com/smart-card-threats. html. 1999.

Schneier, Bruce. "Attack trees." Dr. Dobb's journal 24.12 (1999): 21-29.

Secunia Half Year Report 2013, http://secunia.com/company/2013_yearly_report_brief/

Sheyner, Oleg, and Jeannette Wing. "Tools for generating and analyzing attack graphs." Formal methods for components and objects. Springer Berlin Heidelberg, 2004.

Sheyner, Oleg, et al. "Automated generation and analysis of attack graphs." Security and Privacy, 2002. Proceedings. 2002 IEEE Symposium on. IEEE, 2002.

Singhal, Anoop, and Xinming Ou. " Techniques for enterprise network security metrics." Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies. ACM, 2009.

Steven Templeton and Karl Levitt. A requires/provides a model for computer attacks. In Proceedings of the New Security Paradigms Workshop, Cork, Ireland, 2000.**Steven, Noel, and Sushil Jajodia.** "Understanding complex network attack graphs through clustered adjacency matrices." Computer Security Applications Conference, 21st Annual. IEEE, 2005.**Trifunovic, Aleksandar, and William J. Knottenbelt.** "Parkway 2.0: A parallel multilevel hypergraph partitioning tool." Computer and Information Sciences-ISCIS 2004. Springer Berlin Heidelberg, 2004. 789-800.

Vesely, W.E., Goldberg, F.F., Roberts, N., Haasl, D.: Fault Tree Handbook. Technical Report NUREG-0492, U.S. Regulatory Commission (1981).

Williams, Leevar, Richard Lippmann, and Kyle Ingols. "An interactive attack graph cascade and reachability display." VizSEC 2007. Springer Berlin Heidelberg, 2008. 221-236.

Y. Zhu, S. Ye, and X. Li. Distributed PageRank computation based on iterative aggregation-disaggregation methods. In Proc. 14th ACM Conf. Info. and Knowledge Management, pages 578–585, 2005.

Zonghua, Shuzhen Wang, and Youki Kadobayashi. "Exploring attack graph for cost-benefit security hardening: A probabilistic approach." Computers & Security (2012).

BIBLIOGRAPHY

Aven, Terje. "A unified framework for risk and vulnerability analysis covering both safety and security." Reliability Engineering & System Safety 92.6 (2007): 745-754.

Bace, Rebecca, and Peter Mell. NIST special publication on intrusion detection systems. BOOZ-ALLEN AND HAMILTON INC MCLEAN VA, 2001.

Bradley, Jeremy T., et al. "Hypergraph partitioning for faster parallel pagerank computation." Formal Techniques for Computer Systems and Business Processes. Springer Berlin Heidelberg, 2005. 155-171.

Brin, Sergey, and Lawrence Page. "The anatomy of a large-scale hypertextual Web search engine." Computer networks and ISDN systems 30.1 (1998): 107-117.

Catalyurek, Umit V., and Cevdet Aykanat. "Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication." Parallel and Distributed Systems, IEEE Transactions on 10.7 (1999): 673-693.

Çatalyürek, Ümit V., and Cevdet Aykanat. "A Fine-Grain Hypergraph Model for 2D Decomposition of Sparse Matrices." IPDPS. Vol. 1. 2001.

Chinchani, Ramkumar, et al. "Towards a theory of insider threat assessment." Dependable Systems and Networks, 2005. DSN 2005.

Cremonini, Marco, and Dmitri Nizovtsev. "Understanding and influencing attackers' decisions: Implications for security investment strategies." (2006).

Dacier, Marc, Yves Deswarte, and Mohamed Kaâniche. "Models and tools for quantitative assessment of operational security." (1996): 177.

Del Rosario, Juan Miguel, Rajesh Bordawekar, and Alok Choudhary. "Improved parallel I/O via a two-phase run-time access strategy." ACM SIGARCH Computer Architecture News 21.5 (1993): 31-38.

Evans, Shelby, et al. "Risk-based systems security engineering: Stopping attacks with intention." Security & Privacy, IEEE 2.6 (2004): 59-62.

Halfond, William GJ, and Alessandro Orso. "Preventing SQL injection attcks using AMNESIA." Proceedings of the 28th international conference on Software engineering. ACM, 2006. Ii Xiaohu - A Stochastic Model for Quantitative Security Analyses of Networked Systems, 2011.

Hamid, Thaier, and Carsten Maple. "A Graph theoretical approach to Network Vulnerability Analysis and Countermeasures." (2011).

Hamid, Thaier, Carsten Maple, and Paul Sant. "Methodologies to develop quantitative risk evaluation metrics." (2012).

Hamid, Thaier, Carsten Maple, and Yong Yue. "A parallel algorithm to calculate the CostRank of a network." (2012).

Hamid, Thaier, Carsten Maple. Article: Network Attacks and Countermeasures, International Conference on Education and Information Management (ICEIM-2013), Penang, Malaysia.

Hamid, Thaier, Carsten Maple. Article: Theoretical And Practical Approaches to Develop a Parallel Cost Rank Algorithm of a Network, Symposium on Emerging Trends in Advanced Computing (SETAC-2012) - 15th May 2012.

Haveliwala, Taher H. "Efficient encodings for document ranking vectors." Stanford University Technical Report (2002).

Haveliwala, Taher H. "Topic-sensitive pagerank." Proceedings of the 11th international conference on World Wide Web. ACM, 2002.

Ingols, Kyle, Richard Lippmann, and Keith Piwowarski. "Practical attack graph generation for network defense." Computer Security Applications Conference, 2006. ACSAC'06. 22nd Annual. IEEE, 2006.

Jansen, Wayne. Directions in security metrics research. DIANE Publishing, 2010.

Jensen, Eric. "Computer Attacks on Critical National Infrastructure: A Use of Force Invoking the Right to Self-Defense." Stanford Journal of International Law 38 (2002): 207.

Kamvar, Sepandar, et al. "Exploiting the block structure of the web for computing pagerank." Stanford University Technical Report (2003).

Karypis, George, Kirk Schloegel, and Vipin Kumar. "Parmetis: Parallel graph partitioning and sparse matrix ordering library." Version 1.0, Dept. of Computer Science, University of Minnesota (1997).

Kleinberg, Jon M. "Authoritative sources in a hyperlinked environment." Journal of the ACM (JACM) 46.5 (1999): 604-632.

Mirkovic, Jelena, and Peter Reiher. "A taxonomy of DDoS attack and DDoS defense mechanisms." ACM SIGCOMM Computer Communication Review 34.2 (2004): 39-53.

Mukerji, Tapan, et al. "Statistical rock physics: Combining rock physics, information theory, and geostatistics to reduce uncertainty in seismic reservoir characterization." The Leading Edge 20.3 (2001): 313-319.

Mukherjee, Biswanath, L. Todd Heberlein, and Karl N. Levitt. "Network intrusion detection." Network, IEEE 8.3 (1994): 26-41.

Myagmar, Suvda, Adam J. Lee, and William Yurcik. "Threat modeling as a basis for security requirements." Symposium on Requirements Engineering for Information Security (SREIS). 2005.

Noel, Steven, and Sushil Jajodia. "Managing attack graph complexity through visual hierarchical aggregation." Proceedings of the 2004 ACM workshop on Visualization and data mining for computer security. ACM, 2004.

Noel, Steven, and Sushil Jajodia. "Proactive intrusion prevention and response via attack graphs." Practical Intrusion Detection (2009).

Oladimeji, Ebenezer Akin, and Lawrence Adviser-Chung. A policy-based framework for engineering security, privacy, and trustworthiness requirements. University of Texas at Dallas, 2011.

Python, January. "Python Programming Language." Python (programming language) 1 CPython 13 Python Software Foundation 15 (2009): 1.

Ramakrishnan, C. R., and R. Sekar. "Model-based analysis of configuration vulnerabilities." Journal of Computer Security 10.1 (2002): 189-209.

Reisig, Wolfgang, and Grzegorz Rozenberg, eds. Lectures on Petri Nets I: Basic Models: Advances in Petri Nets. Vol. 149. Springer, 1998.

Ritchey, Ronald W., and Paul Ammann. "Using model checking to analyze network vulnerabilities." Security and Privacy, 2000. S&P 2000. Proceedings. 2000 IEEE Symposium on. IEEE, 2000.

Saini, Vineet, Qiang Duan, and Vamsi Paruchuri. "Threat modeling using attack trees." Journal of Computing Sciences in Colleges 23.4 (2008): 124-131.

Schechter, Stuart E. "Toward econometric models of the security risk from remote attacks." Security & Privacy, IEEE 3.1 (2005): 40-44.

Tenable network security, http://www.tenable.com/plugins/index.php?view=newest.

Penang, Malaysia. Tracy, Miles, Wayne Jansen, and Mark McLarnon. "Guidelines on Securing Public Web Servers Web Servers." NIST Special Publication 800 (2002): 44.

Trifunovic, Aleksandar. Parallel algorithms for hypergraph partitioning. Diss. University of London, 2006.

Uçar, Bora, Ümit V. Çatalyürek, and Cevdet Aykanat. "A matrix partitioning interface to PaToH in MATLAB." Parallel Computing 36.5 (2010): 254-272.

Wright, Jason L., Miles McQueen, and Lawrence Wellman. "Analyses of two end-user software vulnerability exposure metrics (extended version) 5." (2013).

Wu, Ximing. "Calculation of maximum entropy densities with application to income distribution." Journal of Econometrics 115.2 (2003): 347-354.

Zaki, Mohammed J., et al. "Parallel algorithms for discovery of association rules." Data Mining and Knowledge Discovery 1.4 (1997): 343-373.

Fruhwirth, Christian, and Tomi Mannisto. "Improving CVSS-based vulnerability" Prioritization and response with context information." Proceedings of the 2009 3rd international Symposium on Empirical Software Engineering and Measurement. IEEE Computer Society, 2009.

APPENDIX A. TOP 10 APPLICATION SECURITY RISKS AND COUNTERMEASURES

In this work, the top ten security risks and countermeasures are presented, adapted from the new version announcement of the OWASP [Williams, Jeff, and Dave Wichers, 2010] Top 10 risks results to calculate the ranks and impotence risks of application security. By using the methodologies and the approaches in this work, this will help to develop plans to think outside the 10 risks and their countermeasures to avoid the impacts of different risks and vulnerabilities existing in their applications and businesses and the optimal method to solve the security problems.

1.Injection: Injection flaws, this type of attack can occur when the attackers send (inject) untrustworthy text/data to, for example, LDAP, SQL queries or OS injection, the injection source data will have a code to exploit the syntax vulnerability of the directed interpreter and might result in account takeover, retrieving data/information, data loss/corruptions or denial of access.

Exploitability	Hackers send (inject) untrustworthy text/data originated attacks that abuse the syntax of the focussed interpreter. Nearly all bases of data could be an injection route.
Occurrence	Happens when the attacker/assailant transmits simple untrustworthy data to Lightight Directory Access Protocol (LDAP), SQL queries, OS, or programming command parameters. The discovery of Injection flaws are simple, in the case of code investigations, but extra complicated by means of testing.
Impact	The impact of Injection flow, in terms of data, alteration/damage/loss, denial of access or account/host takeover.

COUNTERMEASURES:

A. The ideal injection flows countermeasures choice is to evade using the interpreter directly, in its place using a safe Application programming interface (API); such interface will help to provide a location (parameterised interface) for code investigation.
B. In case the API is not on hand, the users should escape using special characters by designing particular interpreter escape syntax.
C. Using whitelist input inspection/validation would aid injection flows protections. The only problem with this method is that many applications require using special characters in their input. Open Web Application Security Project (OWASP), for example, uses a rich whitelist library for their input inspection/validation practice.

Paradigm of injection flow attack: In the following SQL queries call, the un-trusted data used in the structure of vulnerable SQL identify.

```
String query = "SELECT * FROM accounts WHERE custID='" + request.getParameter("id") +"'";
```

The assailant changes the ‘id’ in the getParameter to '1'='1 in their Internet browser to post. The action will alter the significance of the query to display/return all the records account from the database, the assailant uses these vulnerabilities to call up exceptional actions in the database, tolerating an entire host takeover of the database.

2.Cross Site Scripting (XSS): The flaws happen when unreliable data send to a web browser with no appropriate validation/inspection from a submission (an application). XSS permits the hackers to perform arbitrary scripts in intentions web browser, in terms of taking control of user sessions, spoiling web sites, or forwarding the browsing to malicious web sites.

Exploitability	Hackers send untrustworthy text scripts (codes) that exploit the browser interpreter's vulnerability. More or less any basis of data could be attack parameter, counting in-house resources.
Occurrence	XSS is mainly general web application security flaw. The XSS flaws happen when un-reliable data are sent to a web page with no appropriate validation/inspection of the content from a submission (an application). The main three types of XSS flaws are: 1) DOM, 2) Stored, 3) Reflected founded XSS.
Impact	Hackers can execute arbitrary codes (script) in the target host browser, in term of take control on user sessions, spoil web sites, or forward the browsing of malicious websites, insertion of intimidating content, , etc.

COUNTERMEASURES

A. The ideal choice is to accurately get away from the entire untrusted data founded on the HTML background, such as Java scripts, Cascading Styles Sheets (CSS), HTML attributes, Body, which the data needs to be located inside. The developers have to incorporate the filtering/escaping script inside the applications if not using User interface (UI).

B. Using whitelist input inspection/validation with suitable decryptions would support protections against XSS flaws. The main problem with this method is many applications require using special characters in their input. The inspections/validation process includes 1) decrypt the encrypted input. 2) Checking the input length, format, and characters. 3) Implement any other rules essential to accept the input.

3.Broken Authentication and Session Management: Software application utilities/functions to control and manage the sessions and the authentications are frequently not configured correctly, permitting the compromise of accounts, passwords, session's indications, keys, by the hackers, or exploiting the vulnerabilities in the applications to deduce the users' information/authentications.

Exploitability	Software application functions to control and manage the sessions and the authentications are frequently not configured right, let to compromise accounts, password, session's indications, keys, by the hackers.
Occurrence	Application developers often design traditional sessions and authentication methods, the perfect design doesn't exist and hard to achieve, as consequences the applications often comes with vulnerabilities in the authentication process, remember my style, timeout, secret questions and answers..etc. Find mitigations to such vulnerabilities could be now and then is hard, as each application is sole.
Impact	These vulnerabilities could let the hackers to compromise several or all accounts. On one occasion of successful attack, the hackers could perform any task the account holders can do; the accounts with root/admin privilege tolerate more harm.

COUNTERMEASURES

A. The developers should design the application's security controls according to OWASP Application Security Verification Standard (ASKS), in terms of the session management and authentication.
B. A template of application security controls should be used by the programmers/developers with a straightforward API. For example, OWASP Enterprise Security API (ESAKI) is a free, open source template, which enables the developers to design low risk User APIs and Authenticator.

C. As XSS vulnerabilities could be used to take control of user sessions, spoil web sites, or forward the browsing of malicious websites, concerted efforts, in terms of finding countermeasures and mitigations, should be used to evade XSS.

4.Insecure Direct Object References: this type of attack happens when the programmers/developers make a specific location/object (such as database item/key, directory, or file) inside an application available to access without using relevant protections. These items could be controlled, manipulated, and used to gain access to illicit data.

Exploitability	The insiders/attackers, officially considered as legal users of a system, can alter the constraint values related to an object to another object, which they are not officially allowed to access.
Occurrence	The application developers often use the real object's key or name of applications during the web page design. These applications do not always confirm that a specific user using the object is officially authorised, that will result in an insecure direct access flaw for the target object. Examiners/testers can simply use constraint values to sense, discover such types of flaw, and preform a scripts/codes exploration, which illustrates if authorisation is used and verified correctly.
Impact	The insecure direct object references flaws could make all the data oriented by the parameters compromised by attackers. Except if, the namespace is sparse.

COUNTERMEASURES

A. Custom indirect object references. This represents a methodology to avoid direct access of unauthorised properties. For example, if the users have a drop-down menu list with five items, the default method to use the numerical values from one to five to specify the user selection. The indirect method could use the real database key mapped to the pre-user reference. The programmers/developers can use random and sequential access mapping to avoid direct referencing to the objects.
B. Access is validating. All users with a direct object reference from untrustworthy resources must be validated using an access control to guarantee that the user who demanded the object is authorised.

5. Cross Site Request Forgery (CSRF): in this flaw, the attacker forces the target's (victim) vulnerable web browser to direct a fake HTTP call, this results in sending authentication data, cookies of the target's sessions, to a specific web application. The CSRF flaw services the targeted web browser to create a call to vulnerable browser deliberates are valid call from the target.

Exploitability	The CSRF flaws generate a fake HTTP call and direct the target's browser to submit the data through XSS, image labels or many other methods. In the case, if the user is legitimate, the attack will be successful.
Occurrence	As web browsers send identifications routinely such as authentication data and session cookies, hackers can make malicious web pages that generate fake calls that are indistinct from genuine ones.
Impact	If this flaw has been utilised by the attackers, they can alter any data that the legitimate user can change or achieve any task on a target machine the legitimate user can do.

COUNTERMEASURES

Avoiding CSRF needs the exclusion of any unexpected token per capita transaction. This requires creating a unique token; the tokens should, as far as possible, be unique for each user session, but could likewise be unique for every call (request).

A. The ideal selection is to compromise and send the unique token in a hidden field in the body of the HTTP call. If the unique token is present in the URL or URL parameters, such action will increase the risk of token exposure and compromise.

6. **Security Misconfiguration:** All operating systems' platforms and software applications such as web servers and browsers, applications clients and servers should have security configurations and settings to secure different tasks and transactions. These settings and configurations need to be well defined, applied, and preserved, as many software applications are dispatched without default security settings.

Exploitability	The attacker gains access through factory default accounts, vulnerabilities without proper security patches, and many other flaws to advance illegal access to the data and information of an organization or a system.
Occurrence	Security misconfiguration could occur at all operating systems' platforms and software applications such as web servers and browsers, applications clients and servers. Network and security administrators and developers/programmers must work closely to make sure that all the software applications and operating systems are configured correctly in terms of security. Auto-security scanners should be used to detect

	different vulnerabilities and omitted security patches, use of default factory accounts, misconfigurations, etc.
Impact	Security misconfiguration often gives attackers illegal access to the data and information and utilities of the system. Sometimes, Security misconfiguration causes a comprehensive system take over.

COUNTERMEASURES

A. The security configuration process should be created in a reliable and stress-free form and should be fast to implement in similar settings. The configuration should be automated through deployment applications to reduce the effort needed to secure different systems.
B. The process should maintain operating systems, software applications, and security patches updated in timely methods for all systems.
C. The network design, architecture should afford proper security between different components in the network. Auto-security scanners should be used to detect different vulnerabilities and missing security patches

6. **Failure to Restrict URL Access**: a lot of web applications verify the access privileges/rights before proceeding to the protected links; all web applications should do analogous access rights checking before proceeding to a different link. Sometimes the attackers will be talented enough to create a fake URL to access protected links.

Exploitability	The insider system users can amend the URL to gain access to the protected web, if web applications do not verify the access privileges. In the same way, unspecified users (anonymous) can access protected or private web pages.
Occurrence	Some web applications do not verify access rights before proceeding to a protected link. On occasion, in some web applications, the URL security is handled through settings and configurations. The occurrence of this type of flaw might happen due to the misconfigurations. The programmers/developers should impose a suitable code to detect such flaws.
Impact	Failure to restrict URL access lets attackers gain access to illegitimate tasks. Root/Administrative tasks are the main objectives for the attackers in this category.

COUNTERMEASURES

Avoiding unofficial URL access requires appropriate verification and appropriate permission for each web page.

A. Enforcing appropriate verification and permissions policies of each web page is the key solution to this type of flaw.
B. In order to reduce the complications of the policies' execution, the policies must be easy and well defined to be configured.
C. The impose method(s) must reject all access in the first instance, then grant privileges and permissions to different users to gain access to each web page
D. In the case where the web pages need to forward the link to a protected web page, the developers must be sure to use appropriate conditions to verify the privileges and the permissions.

8.Unvalidated Redirects and Forwards: Often the web applications forward and pass on users to other links using untrustworthy data to find out the target websites. The lack of appropriate verifications and confirmations will let the attackers forward the links to malware or phishing web pages and might lead to access protected web pages.

Exploitability	An attacker could forward or redirect users to malware of unsafe links due to the lack of appropriate verifications and confirmations.
Occurrence	Often the web applications forward and pass on users to other links using untrustworthy data to find out the target websites. Occasionally the goal web page is adapted inside an unauthorised constraint, permitting attackers to select the target web page. The lack of appropriate verifications and confirmations will let the attackers forward the links to malware or phishing web pages, and might lead to access protected web pages.
Impact	Invalidated redirects and forwards flaws might lead to password disclosure, installing malware, and hacking utilities or go around access verification process.

COUNTERMEASURES

A. Evade using forwards and pass on.
B. In the case of using forward and redirect, the use of target constraints (parameters) in the process of finding out the target websites should be avoided.
C. If the use of target parameters cannot be avoided, the parameters' values should be verified and the user's privileges and permissions must be checked.
D. Using ESAPI to countermand the sendRedirect() process in the web application to ensure that all the forwarding and the redirecting to different

links and websites are secure and safe. It is suggested and helpful to use plotting (mapping) values if targets' constraints have been used, in which case the web server will translate the plotting values to find out the target websites.

9.Insecure Cryptographic Storage: The protection of critical data such as credit and debit card information, Social Security Number (SSNs) and authentication identifications is not properly constructed in numerous web applications using a suitable encryption method. Attackers might utilise this vulnerability to conduct cybercrimes such as identity theft and credit and debit card fraud.

Exploitability	Usually attackers do not break down the encryption methods to decrypt the data streams, instead they use different methodologies such as catch a clear text of encrypted data, determine the encryption key, or access the decrypted data through some channels.
Occurrence	The main cause of this flaw is due to not using any encryption algorithms to protect important or critical data. Other occurrences are when specific encryption methods are used, due to utilising weak encryption algorithms, do not use rotating keys, using unsafe keys or using fragile hashes to secure passwords. When the attackers access the networks as external users, they face difficulties to discover such vulnerabilities because of the access limitations.
Impact	Insecure cryptographic storage flaws occurrences might result in a compromise of all the data in the system, including authentications, credentials, and configuration.

COUNTERMEASURES

A. Security planning for all networks must pay specific attention to using appropriate encryption algorithms to secure all sensitive data, special consideration should be taken to internal and external users as attackers.
B. In the backup's procedures, the off-site backups must be encrypted, the backups of encryption keys must be managed individually.
C. Strong encryption algorithms with robust encryption keys should be used in the process of data protections.
D. Robust hashes algorithms should be used to secure passwords.

10.Insufficient Transport Layer Protection: protection of critical network communications has been often unsuccessful because the software applications do not use cryptographic algorithms to encrypt sensitive network, and in the case of using them, sometimes using fragile algorithms, unacceptable or expired certificates or using them in improper ways.

Where transport layer security is implemented, that will impact the network segment design. In this case, the Secure Socket Security (SSL) /Transport Layer Security (TLS) needed for all pages in the segment, the SSL/TLS implementation will affect the network performance. For that reason, sometimes the use of SSL/TLS for critical pages only might lead to compromise data, sessions IDs.

Exploitability	Observing network traffic from a specific user could be problematic, especially when the monitoring of the network traffic happened during the users accessing or communicating with vulnerable segments.
Occurrence	Most of the applications do not secure the network traffic; they use SSL/TLS through the authentication process, and frequently do not properly protect network traffic. Usually, they use SSL/TLS through authentication, but not elsewhere, showing totally

	transported data including session IDs. Applications using unacceptable or expired certificates or using them in improper ways.
Impact	Insufficient transport layer protection flaws could compromise users' data, account robbery. In the case that root/administrator account using the system, the entire system might be compromised. The bad SSL setting could lead to phishing attacks.

COUNTERMEASURES

A. Ensure the SSL/TLS or any other reliable technologies are used for back and front end and all additional links.
B. Ensure SSL configured for all designated web pages and not SSL designated to specific pages that forward to the SSL web page.
C. Ensure that the selected SSL/TLS supplier only backing robust algorithms.
D. Ensure that all critical cookies have secure flag setting.
E. Guarantee that certificate is usable, acceptable, or not expired.

APPENDIX B.COMMON VULNERABILITY SCORING SYSTEM (CVSS) STRUCTURE

The commercial open framework CVSS is intended to indicate the impact‘s numerical severity quantitative measures of different vulnerabilities in terms of the Base, Temporal, and Environmental metrics to help in finding a specific mitigation of the vulnerability and to set the priority of response.

The formulae used to calculate the impact‘s numerical severity quantitative measures of CVSS were considered to be accurate, inclusive and straightforward to employ. For these reasons, CVSS was widely used to perform a reality testing of vulnerabilities in the end user background.

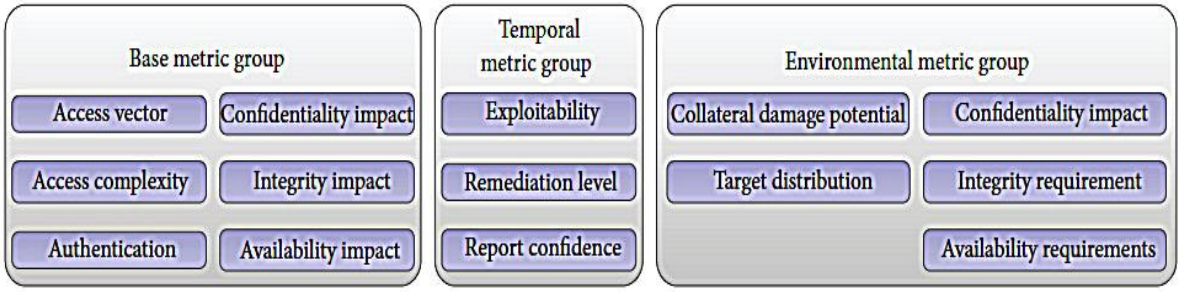


Figure B-1. The CVSS Model shows the relations between Base, Temporal, and Environmental Metric Groups modified from [Mell et al., 2007]

As mentioned earlier, CVSS is comprised of three metric sets: Base, Temporal, and Environmental metrics; respectively containing established numerical severity quantitative measures aimed to provide the end user with a complete combination mark to represent the risk and severity of a vulnerability.

A. **Base Metric collection** [Mell et al., 2007] have general details and qualities of a given vulnerability that do not change if the vulnerability exists in different environments or over time, base metrics gives the general severity of a given vulnerability.

It is represented by two subdivisions:

- Exploitability: represented using Authentication, Access Complexity, Access Vector;
- Impact: represented using Integrity, Availability, and Confidentiality.

Access Vector (AV): weighs the access methodologies used to exploit a vulnerability an attacker can reach to exploit vulnerability in terms of vulnerabilities access (such as locally, network) and if configuration settings are essential to exploit them in term of **access.**

Access vector encompasses the following factors:

a. Local (L): The attackers can access and exploit vulnerabilities in a LAN locally, there is no need for any network connections,
b. Adjacent Network (A): The hackers can access the collision or broadcast domain of the application software where the vulnerability is based.
c. Network (N): To exploit a vulnerability, the hackers do not need to access a LAN or a neighbouring network, as the vulnerabilities based on the application software are preordained for network stack and could be accessed through Internet access only.

Access Complexity (AC): AC measures how complicated an attack is, designed to measure the level of difficulties of an attack to exploit the vulnerability in the aimed target.

a.High (H): represents the high risk in terms of access complexity, i.e. the attacker has considerable talent and experience to compromise further systems in a multi-step attack process, or using advanced social engineering techniques, which might lead that attacker to gain root privileges. An example of such attack, the Injection flaws; this type of attack can occur when the attackers sends (injects) untrustworthy text/data to, for example, LDAP, SQL queries or OS injection, the injection source data will have a code to exploit the syntax vulnerability of the directed interpreter and might result in account takeover.

b.Medium (M): The access complexity is to some extent skilled; for instance, only various users are capable of completing specific attacks on some explicit systems, the concerned configuration changes are special and require an acceptable talent, or specific information collection is demanding.

c.Low (L): represents a low risk in terms of accessible environments, for example the affected applications or systems need admission to a variety of users, the attackers require a small amount of talent or the information collection needed is less.

Authentication (Au): In order for attackers to exploit a vulnerability, they might or might not have to authenticate, Au classifies the number of times an attacker need to authenticate in order to conduct a specific attack.

a.Multiple (M): The attacker needs authentication twice or more (even if the same authentication has been used) to exploit.

b.Single (S): The attacker needs a single authentication to exploit a vulnerability in the target system.

c.None (N): The attacker does not need any authentication to exploit a vulnerability in the target system. An example of such attack, Cross Site Scripting attack, whereby the flaws happen when unreliable data is sent to a web browser with no appropriate validation/inspection from a submission (an application). XSS permits the hackers to perform arbitrary scripts in intentions web browser, in terms of taking control of user sessions, spoiling web sites, or forwarding the browsing to malicious web sites.

Confidentiality Impact (CI): An attacker might compromise the confidentiality of the data (intentional data disclosures) in a system or in a communication channel between sender and recipient. CI evaluates the consequence and harm on confidentiality if the attacker succeeds in exploiting a vulnerability. An example of such attack, the insecure direct object references flaws could make all the data oriented by the parameters compromised by attackers.

a. Complete (C): Entire data being compromised, data disclosures happened during the exploitation of a vulnerability.

b. Partial (P): Significant data disclosures happened during the exploitation of a vulnerability

c. None (N): If there is no harm or consequence on the confidentiality during the exploitation of a vulnerability.

Integrity Impact (Inl): An attacker might compromise the consistency and the accuracy of the data in a system; the INI evaluates the consequence and harm on integrity if the attacker succeeds to compromise the integrity of a system by exploiting a specific vulnerability. An example of such attack, the Cross Site Request Forgery, the attackers can alter any data that the legitimate user can change or achieve any task on a target machine the legitimate user can do.

a. Complete (C): Entire data consistency and the accuracy impact of a system; affecting the integrity during the exploitation of a vulnerability of systems.
b. Partial (P): Significant data consistency and the accuracy impact affecting the integrity during the exploitation of a vulnerability of a system.
c. None (N): If there is no harm or consequence affecting the integrity during the exploitation of a vulnerability.

Availability Impact (AI): An attacker might compromise the availability of a system in terms that an authorised user can access resources in a consistent and appropriate way, AI evaluates the impact of exploiting a specific vulnerability on the availability of a system. An example of such attack, exploiting the vulnerability CVE-2009-1758 in Linux kernel by local user with no privilege, which makes the kernel, crash and bring down the whole system.

a. Complete (C): Completely affect the system availability, such as system shutdown.

b. Partial (P): Instabilities in resources and performance reduction, which might affect the system availability partially.
c. None (N): If there is no harm or consequence affecting the availability during the exploitation of a vulnerability.

The representation of Base Metrics in the following format:

AV:	L	AC:	H	Au:	M	C:	N	I:	N	A:	N
	A		M		S		P		P		P
	N		L		N		C		C		C

OR:

(AV: [L, A, N]/ AC: [H ,M ,L]/ Au: [M ,S ,N]/C: [N ,P ,C]/I: [N ,P ,C]/A: [N ,P ,C])

As shown in Table 2.1, 2.2 each rating for the CVSS Attributes assigned with the rating value metric

The equation to calculate Base Metric (BS) is:

$$BS = round^{1}((0.6 * I + 0.4 * E_x - 1.5) * Function(I))$$

where round[1]=round to one decimal, I=Impact, E_x=Exploitability.

$$I = 10.41 * (1 - (1 - CI) * (1 - InI) * (1 - AI))$$

$$E_x = 20 * AV * AC * Au$$

where $Function(I) = 0 \quad If \quad I = 0$

$Function(I) = 1.176 \quad If \quad I \neq 0$

CVSS metrics group	CVSS attribute	Rating	Rating value
base metric Exploitability	access required (AV)	local (L)	0.395
		adjacent network (A)	0.646
		network (N)	1.0
	attack complexity (AC)	high (H)	0.35
		medium (M)	0.61
		low (L)	0.71
	authentication instances (Au)	multiple (M)	0.45
		single (S)	0.56
		none (N)	0.704

Table B-1. CVSS base metrics- exploitability [Mell et al.,2007]

CVSS metrics group	CVSS attribute	Rating	Rating value
base metrics impact	confidentiality impact (C)	none (N)	0.0
		partial (P)	0.275
		complete (C)	0.660
	integrity impact (I)	none (N)	0.0
		partial (P)	0.275
		complete (C)	0.660
	availability impact (A)	none (N)	0.0
		partial (P)	0.275
		complete (C)	0.660

Table B-2. CVSS base metrics- Impact [Mell et al., 2007]

B. Temporal Metric collection:

The threat presented by the vulnerability might modify and change over the time of vulnerability.

The temporal metric composes three factors:

- Exploitability
- Remediation Level

•Report Confidence

Because the temporal metrics are optional, so the metrics values of the three factors might have no effect on the final score.

Exploitability (E): Evaluates the existence of exploiting codes and techniques.

1. Unproven (U): The exploits codes/techniques are not available.
2. Proof-of-Concept (POC): Indicates the availability of exploit code/techniques or an impracticable exploit.
3. Functional (F): Indicates the availability of functional exploit code.
4. High (H): There are functional portable independent codes/techniques and details are widely available.
5. Not Defined (ND): The ND score would not affect the temporal Metric score. ND only gives an indication to the formula to ignore this metric.

Remediation Level (RL): Points towards the availability of remediations (mitigations) and performs the numerical rating values to each case.

Official Fix (OF): Indicates if the retailer fix (patches) is available, for instance an authorised upgrade or fixes (patches).

The possible values for this metric are:

1.Temporary Fix (TF): indicates the scales of official temporary fix availability.
2.Workaround (W): indicates whether an unofficial solution is available.
3.Unavailable (U): indicates official fixes, Solutions are not available.
4.Not Defined (ND): No value assigned.

Report Confidence (RC): Indicates the level of trust present of a vulnerability and the degree of reliability of the acknowledged technical information.

The factors' values of this metric are shown below:

1. Unconfirmed (UC): this factor indicates a single unconfirmed report or multiple inconsistence sources.
2. Uncorroborated (UR): several unofficial reports, which might be from research groups or independent security corporations.
3. Confirmed (C): The trouble/vulnerability officially reported to the retailer's own software application or practical exploits script exists in the public domain.
4. Not Defined (ND): there is not enough information on the vulnerability, no rating value is allocated in this case.

The representation of Temporal Metric is in the following format:

E:	U	RL:	OF	RC:	UC
	POC		TF		UR
	F		W		C
	H		U		ND
	ND		ND		

OR:

(E:[U,POC,F,H,ND]/RL:[OF,TF,W,U,ND]/RC:[UC,UR,C,ND])

The Temporal Metric Score (TS) is calculated as follows:

$$\mathbf{TS} = round^{1}\,(\mathbf{B}_S * \mathbf{E} * \mathbf{R}_L * \mathbf{R}_C)$$

CVSS metrics group	CVSS attribute	Rating	Rating value
Temporal metrics	exploitability tools & techniques (E)	unproved (U)	0.395
		proof-of-concept(P)	0.646
		functional (F)	0.95
		high (H)	1.0
	remediation level (RL)	official fix (OF)	0.87
		temporary fix (TF)	0.90
		workaround (W)	0.95
		unavailable (U)	1.0
	report confidence (RC)	unconfirmed (UC)	0.90
		Un corroborative (UR)	0.95
		confirmed (C)	1.0

Table B-3. CVSS temporal metrics [Mell et al., 2007]

C.Environmental Metric collection: indicates the alteration of vulnerability metric that related to the user settings and configurations of a particular asset's environment**.** The three factors that CVSS releases are:

- Collateral Damage Potential
- Target Distribution
- Security Requirements

Collateral Damage Potential (CDP): indicates the possible damage or theft of resources such as theft of equipment and financial damage to profits /production.

The probability values for this metric are:

1. None (N): No possible damage or theft of resources or loss of profits.
2. Low (L): Minor loss of resources or slight financial damage to profits.
3. Low-Medium (LM): Medium loss of resources or average financial/productivity damage.

4. Medium-High (MH): Major loss of resources or significant financial/productivity damage.
5. High (H): Catastrophic loss of resources or catastrophic financial/productivity damage.
6. Not Defined (ND): No metric value will be assigned.

Target Distribution (TD): TD indicates the figure/percentage of the target systems that could be impacted by the vulnerability, according to environment setting.

The possible values for this metric are listed below:

1. None (N): (0%) The target systems are not present, or the systems available are only in specialised or laboratory site.
2. Low (L): (2-25%) targets that could be impacted by the vulnerability.
3. Medium (M): (26-75%) targets that could be impacted by the vulnerability.
4. High (H): High scale of system targets (76-100%) exists.
5. Not Defined (ND): No metric value will be assigned.

Security Requirements (CR, IR, and AR): Grades special score based on the assets/targets significance to the establishment. The security requirements three factors in terms of assets impact are Confidentiality Requirement (CR), Integrity Requirement (IR), and Availability Requirement (AR). These metrics influence the base metric corresponding values of availability, integrity, and confidentiality.

The impact on specific target or system could be:

1.Low (L): Possible limited impact/ effect only.
2.Medium (M): A serious attack consequence on significant assets /targets.
3.High (H): A disastrous attack consequence on significant assets /targets.
4.Not Defined (ND): No metric value will be assigned.

The representation of Environmental Metric is in the following format:

CDP:	N	TD:	N	CR:	L	IR:	L	AR:	L
	L		L		M		M		M
	LM		M		H		H		H
	MH		H		ND		ND		ND
	H		ND						
	ND								

(CDP:[N,L,LM,MH,H,ND]/TD:[N,L,M,H,ND]/CR:[L,M,H,ND]/IR:[L,M,H,ND]/AR:[L,M,H,ND])

The Adjusted Impact Score (AI) is calculated as follows:

$$AI = \text{Minimum}(10, 10.41*(1-CI*CR)*(1-InI*IR)*(1-AI*AR))$$

The Base (BS_E) score is recalculated using Adjusted Impact (AI) instead of Impact as shows below:

$$BS_E = round^1((0.6*AI + 0.4*E_x - 1.5)*Function(I))$$

The Adjusted Temporal Score (AT) is calculated using BS_E as follows:

$$AT = round^1(\mathbf{BS_E * E * R_L * R_C})$$

The Environmental Metric Score (ES) is calculated as follows:

$$ES = round^1(((AT + (10 - AT)*CDP*TD)$$

The formula above shows the methodology to calculate the ES by multiplying the (10 - adjusted temporal) * (represent exploitability, remediation level and report confidence) with CollateralDamagePotential and target distribution (explained earlier in this chapter) and adding the result to the value of adjusted temporal.

CVSS metrics group	CVSS attribute	Rating	Rating value
environmental metrics	confidentiality requirement (CR)	low (L)	0.5
		medium (M)	1.0
		high (H)	1.51
	integrity requirement (IR)	low (L)	0.5
		medium (M)	1.0
		high (H)	1.51
	availability requirement (AR)	low (L)	0.5
		medium (M)	1.0
		high (H)	1.51
	collateral damage potential (CDP)	none (N)	0.0
		low (L)	0.1
		lowmedium (LM)	0.3
		mediumhigh (MH)	0.4
		high (H)	0.5
	Target Distribution (TD)	Non(N)	0.0
		low (L)	0.25
		medium (M)	0.75
		high (H)	1.0
		Not Defined(ND)	1.0

Table B-4. CVSS environmental metrics [Mell et al., 2007]

APPENDIX C. THE LIST OF VULNERABILITIES OBTAINED FROM NESSUS SCANNING

The list of Vulnerabilities is obtained after using Nessus scanning reports is shown below:

Results Details:

Scan Information PC1:
PC1: 192.168.153.160

Vn1:11324 - CVE-2004-0230, TCP/IP packet's sequence number guess, the attacker directs the spoofed RST to the remote host and terminates established connections, causing DoS and partially disrupting the availability.

DVSS Score Medium

Intrinsic =5.0:

AV:	N	AC:	L	Au:	N	C:	N	I:	N	A:	P

Time-based =3.9:

E:	P	RL:	OF	RC:	C

Ecological = 57:

TD:	M	CDP:	MH	CR:	L	IR:	L	AR:	M

$Cost_{Vn1} = 57$

Vn2:46628: CVE-2010-0025, SMTP and Exchange Server services (windows server 2003) might lead to DOS and partially impact the confidentiality.

CVSS Base Score Medium

Intrinsic =5.0:

AV:	N	AC:	L	Au:	N	C:	P	I:	N	A:	N

Temporal=4.1:

E:	F	RL:	OF	RC:	C

Cost $_v$= 47:

TD:	M	CDP:	LM	CR:	L	IR:	L	AR:	M

$Cost_{Vn2} = 47$

Vn3:11467: CVE-1999-1011, This vulnerability in Microsoft IIS Data Access Components (MDAC) RDS, which might permit attackers from remote sites to access Object Linking and Embedding (OLE) and run Illogical scripts.

Execution CVSS Base Score Critical

Intrinsic =10.0:

AV:	N	AC:	L	Au:	N	C:	C	I:	C	A:	C

Time-based 7.8:

E:	P	RL:	OF	RC:	C

Cost $_v$= 89

TD:	H	CDP:	H	CR:	H	IR:	H	AR:	H

$Cost_{Vn3} = 89$

Vn4:12172 - CVE-2002-1142, Buffer overflow vulnerability affects a remote host, Microsoft Data Access Components (MDAC) RDS Data Stub Remote Overflow

CVSS Base Score High

Intrinsic =7.5:

AV:	N	AC:	L	Au:	N	C:	P	I:	P	A:	P

Time-based 7.8:

E:	P	RL:	OF	RC:	C

Cost $_v$= 52

TD:	H	CDP:	MH	CR:	H	IR:	H	AR:	H

$Cost_{Vn4} = 52$

Vn5:12324 - CVE-2010-0386, HTTP TRACE / TRACK vulnerability enables attackers to take cookies and authentication ID using a cross-site tracing attack (XST).

CVSS Base Score Low

Intrinsic =4.3:

AV:	N	AC:	M	Au:	N	C:	P	I:	N	A:	N

Time-based 3.6:

E:	F	RL:	OF	RC:	C

Cost $_v$= 38:

TD:	L	CDP:	L	CR:	M	IR:	L	AR:	L

$$Cost_{Vn5} = 38$$

Vn6:11981 - CVE-2003-0717, In Windows NT up to server 3003, the messaging services do not correctly check the message size, the attackers might utilise this vulnerability by running illogical scripts using buffer overflow and cause the Messenger Service to fail.

CVSS Base Score High

Intrinsic =7.5:

AV:	N	AC:	L	Au:	N	C:	P	I:	P	A:	P

Time-based 5.9:

E:	P	RL:	OF	RC:	C

Cost $_v$= 71:

TD:	M	CDP:	MH	CR:	M	IR:	M	AR:	M

$$Cost_{Vn6} = 71$$

Vn7:21283 - CVE-2006-3439, MS06-040: Vulnerability in Microsoft Server Service that permits the attackers to run remote script and totally control the target system.

CVSS Base Score critical

Intrinsic =10.0:

AV:	N	AC:	L	Au:	N	C:	C	I:	C	A:	C

Time-based 7.8:

E:	F	RL:	OF	RC:	C

Cost $_v$= 90.0:

TD:	ND	CDP:	MH	CR:	H	IR:	H	AR:	H

$Cost_{Vn7} = 90.0$

Vn8:18519: CVE-2005-1983, Vulnerability in Plug and Play Service, permits remote attackers to influence the plug and play services by inserting illogical script using a worm such as Mytob; this could enable the attackers as local users to escalate the privileges.

CVSS Base Score Critical

Intrinsic =10.0:

AV:	N	AC:	L	Au:	N	C:	C	I:	C	A:	C

Time-based 7.8:

E:	P	RL:	OF	RC:	C

Cost $_v$= 87.0:

TD:	U	CDP:	MH	CR:	H	IR:	H	AR:	H

$Cost_{Vn8} = 87.0$

Vn9:21245: CVE-2006-1314, Vulnerability in Server Service that permits the attackers to run remote script and prompts memory corruption and avoids length limitations and partially influence the confidentiality, Integrity and availability.

CVSS Base Score Medium

Intrinsic =7.5:

AV:	N	AC:	L	Au:	N	C:	P	I:	P	A:	P

Time-based 5.9:

E:	P	RL:	OF	RC:	C

Cost $_v$= 60:

TD:	M	CDP:	LM	CR:	M	IR:	L	AR:	L

$Cost_{Vn9} = 60$

Vn10:54321 - CVE-2000-1200, Vulnerability in Microsoft Windows SMB, influences the confidentiality by permitting remote attackers to get the domain SID and have a list of all users in a domain.

CVSS Base Score Medium

Intrinsic =5.0:

AV:	N	AC:	L	Au:	N	C:	P	I:	N	A:	N

Time-based 3.9:

E:	P	RL:	OF	RC:	C

Cost $_v$= 41:

TD:	L	CDP:	L	CR:	M	IR:	L	AR:	L

$Cost_{Vn10} = 41$

Vn11:54322 - Vulnerability in SMB, practising Host SID to Enumerate Local Users without Credentials, pPartially influences the confidentiality

CVSS Base Score Medium

Intrinsic =5.0:

AV:	N	AC:	L	Au:	N	C:	P	I:	N	A:	N

Time-based 3.9:

E:	P	RL:	OF	RC:	C

Cost $_v$= 44:

TD:	L	CDP:	LM	CR:	M	IR:	L	AR:	L

$Cost_{Vn11} = 44$

Scan Information PC2:
IP2: 192.168.153. 31

Vn1:13124 - CVE-2004-0230, TCP/IP packet's sequence number guess, the attacker directs the spoofed RST to the remote host and terminates established connections, causing DoS and partially disturbing the availability.

Risk Factor Medium

Intrinsic =5.0:

AV:	N	AC:	L	Au:	N	C:	N	I:	N	A:	P

Time-based 3.6:

E:	U	RL:	TF	RC:	UC

Cost v= 50.0:

TD:	L	CDP:	MH	CR:	L	IR:	L	AR:	H

$C_{Vn1} = 50.0$

Vn2:55172 - CVE-2011-3188, Linux Kernel TCP Sequence Number Generation Security Weakness

Risk Factor Medium

Intrinsic =6.8:

AV:	N	AC:	M	Au:	N	C:	P	I:	P	A:	P

Time-based 4.9:

E:	U	RL:	TF	RC:	C

Cost v= 64.0:

TD:	M	CDP:	MH	CR:	M	IR:	M	AR:	M

$C_{Vn2} = 64.0$

Vn3:51452 – CVE-2010-3867, ProFTPD: Multiple Vulnerabilities

Risk Factor Critical

Intrinsic =10.0:

AV:	N	AC:	L	Au:	N	C:	C	I:	C	A:	C

Time-based 8.7

E:	H	RL:	OF	RC:	C

Cost v= 92.0:

TD:	H	CDP:	MH	CR:	H	IR:	H	AR:	H

$C_{Vn3} = 92.0$

Vn4:52475 - CVE-2010-0110, ProFTPD: SQL Buffer Overflow

Risk Factor Critical

Intrinsic =10.0:

AV:	N	AC:	L	Au:	N	C:	C	I:	C	A:	C

Time-based 8.7:

E:	H	RL:	OF	RC:	C

Cost $_v$= 91.0:

TD:	M	CDP:	MH	CR:	H	IR:	H	AR:	H

$C_{Vn4} = 91.0$

Vn5:12658 – CVE-2003-0831, Pro FTPD File Transfer Newline Character Overflow

Risk Factor Critical

Intrinsic = 9.1:

AV:	N	AC:	L	Au:	S	C:	C	I:	C	A:	C

Time-based 7.9:

E:	H	RL:	OF	RC:	C

Cost $_v$= 90.0:

TD:	N	CDP:	H	CR:	H	IR:	H	AR:	H

$C_{Vn5} = 90.0$

Vn6:14568 – CVE-2005-4816, Pro FTPD Buffer overflow, permits attackers to impact the availability of the system (crash the systems) using DoS, permits the attackers to run remote password script.

Risk Factor Medium

Intrinsic =7.5:

AV:	N	AC:	L	Au:	N	C:	P	I:	P	A:	P

Time-based 5.5:

E:	U	RL:	OF	RC:	C

Cost $_v$= 69.0:

TD:	M	CDP:	MH	CR:	M	IR:	M	AR:	M

$C_{Vn6} = 69.0$

Vn7:46126 - CVE-2009-3555, Apache 2.2 less than 2.2.15 Multiple Vulnerabilities

Risk Factor Medium

Intrinsic = 5.7:

AV:	N	AC:	M	Au:	N	C:	N	I:	P	A:	P

Time-based 5.0:

E:	H	RL:	OF	RC:	C

Cost $_v$= 54.0:

TD:	K	CDP:	ML	CR:	L	IR:	M	AR:	M

$C_{Vn7} = 54.0$

Vn8:43164 - CVE-2009-2699, Apache HTTP Server earlier 2.2.14 and additional services do not correctly function with errors, which permits attackers to cause a denial of service remotely through HTTP requests.

Risk Factor: Medium

Intrinsic = 5.0:

AV:	N	AC:	L	Au:	N	C:	N	I:	N	A:	P

Time-based 4.4:

E:	H	RL:	OF	RC:	C

Cost $_v$= 48.0:

TD:	L	CDP:	ML	CR:	L	IR:	L	AR:	M

$C_{Vn8} = 48.0$

Vn9: 55895 - CVE-2011-3192, Apache HTTP Server Byte Range DoS, permits attackers to cause a denial of service remotely and gain privileges.

Risk Factor Critical

Intrinsic =7.8:

AV:	N	AC:	L	Au:	N	C:	N	I:	N	A:	C

Time-based 5.8:

E:	U	RL:	OF	RC:	C

Cost $_v$= 82.0:

TD:	M	CDP:	MH	CR:	H	IR:	M	AR:	M

$C_{Vn9} = 82.0$

Vn10:18582 - CVE-2007-1741, Apache mod_suexec Multiple Privilege Escalation Vulnerabilities.

Risk Factor High

Intrinsic =6.2:

AV:	L	AC:	H	Au:	N	C:	C	I:	C	A:	C

Time-based 5.4:

E:	H	RL:	OF	RC:	C

Cost $_v$= 77.0:

TD:	H	CDP:	H	CR:	H	IR:	M	AR:	H

$C_{Vn10} = 77.0$

Vn11:18573 - CVE-2012-1708, Apache mod_negotiation Multi-Line Filename Upload Vulnerabilities, permits attackers to impact the integrity of the system

Risk Factor Medium

Intrinsic =4.3:

AV:	N	AC:	M	Au:	N	C:	N	I:	P	A:	N

Time-based 3.7:

E:	H	RL:	OF	RC:	C

Cost $_v$= 51.0:

TD:	M	CDP:	ML	CR:	L	IR:	M	AR:	L

$C_{Vn11} = 51.0$

Vn12: 57424 - CVE-2011-3348, Apache 2.2 less than 2.2.21 mod_proxy_ajp, permits attackers to cause a denial of service remotely through HTTP requests.

Risk Factor: Low

Intrinsic =4.3:

AV:	N	AC:	M	Au:	N	C:	N	I:	N	A:	P

Time-based 3.7:

E:	H	RL:	OF	RC:	C

Cost $_v$= 18.0:

TD:	N	CDP:	LM	CR:	L	IR:	L	AR:	M

$C_{Vn12} = 18.0$

Vn13:12746 - CVE-2003-0310, Cross-site scripting (XSS) vulnerability in article view.php for eZ publish 2.2 permits attackers to remotely insert illogical web script.

Risk Factor: Medium

Intrinsic = 5.7:

AV:	N	AC:	M	Au:	N	C:	P	I:	P	A:	N

Time-based 5.0:

E:	H	RL:	OF	RC:	C

Cost $_v$= 54.0:

TD:	L	CDP:	LM	CR:	M	IR:	M	AR:	L

$C_{Vn13} = 54.0$

Vn14:18753 - CVE-2009- 4484, MySQL less than 5.0.90 / 5.1.43 / 5.5.0-m2 Multiple Buffer Overflows

Risk Factor Medium

Intrinsic =7.5:

AV:	N	AC:	L	Au:	N	C:	P	I:	P	A:	P

Time-based 6.5:

E:	H	RL:	OF	RC:	C

Cost $_v$= 65.0:

TD:	M	CDP:	ND	CR:	M	IR:	M	AR:	M

$C_{Vn14} = 65.0$

Vn15:47620 - CVE-2010-1850 MySQL Community Server less than 5.1.47 / 5.0.91 Multiple Vulnerabilities permit remote authenticated attackers to run illogical script through a COM_FIELD_LIST command.

Risk Factor: Medium

Intrinsic =6:

AV:	N	AC:	M	Au:	S	C:	P	I:	P	A:	P

Time-based 5.2:

E:	H	RL:	OF	RC:	C

Cost $_v$= 66.0:

TD:	M	CDP:	MH	CR:	M	IR:	M	AR:	M

$C_{Vn15} = 66.0$

Vn16:46895 - CVE-2010-3676, MySQL Community Server less than 5.1.49 Multiple Vulnerabilities

Risk Factor Low

Intrinsic =4.0:

AV:	N	AC:	L	Au:	S	C:	N	I:	N	A:	P

Time-based 3.3:

E:	F	RL:	OF	RC:	C

Cost $_v$= 38.0:

TD:	L	CDP:	LM	CR:	L	IR:	L	AR:	M

$C_{Vn16} = 38.0$

Vn17:48285 - CVE-2010-2008, MySQL Community Server less than 5.1.48 Denial of Service

Risk Factor Low

Intrinsic =3.5:

AV:	N	AC:	M	Au:	S	C:	N	I:	N	A:	P

Time-based 3.0:

E:	H	RL:	OF	RC:	C

Cost $_v$= 23.0:

TD:	L	CDP:	L	CR:	L	IR:	L	AR:	M

$C_{Vn17} = 23.0$

APPENDIX D. PARALLEL COSTRANK EXPERIMENTS

In this Appendix, a description of our approach and experiments of parallel CostRank presented in Chapter-5.

Sparse matrix decomposition implemented on a link structure of Stanford.edu domain crawl, which contains 281903 pages and 2.3 million links.

Our approach to shorten the size of web crawl is using the MapReduce [Lattanzi, Silvio, et al., 2011] method for adequately compressed graphs.

The MapReduce filtering method using a minimum cut of the graph and reduced the estimated number of vertexes and edges to modify the link data composed as shown in Figure D-1.

The MapReduce algorithm processes the input information, which should store in the form of Key, value. The calculation continues with sequences, which involve three phases of map, shuffle, and cut back.

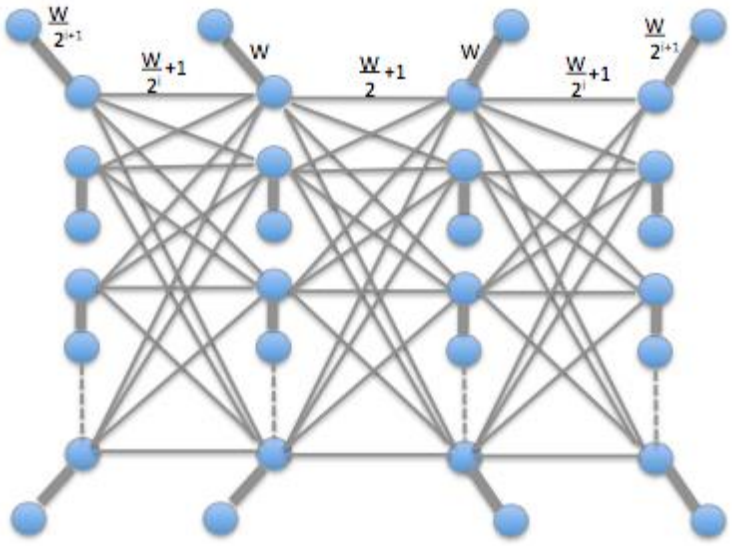


Figure D-1. The subgraph drawn only the edges of weight W, (W/2)+1, ($W/2^{i+1}$) and ($W/2^{i}$)+1 taken from [Lattanzi, Silvio, et al., 2011]

The modified link data decomposed to Stanford-1 with 0.5M links and 61283 pages, Stanford-2 with 1.5M links and 183849 pages, Stanford-3 with 1.725M links and 214491 pages, Stanford-4 with 2.3M links and 281903 pages, as exhibited in Figure D-2.

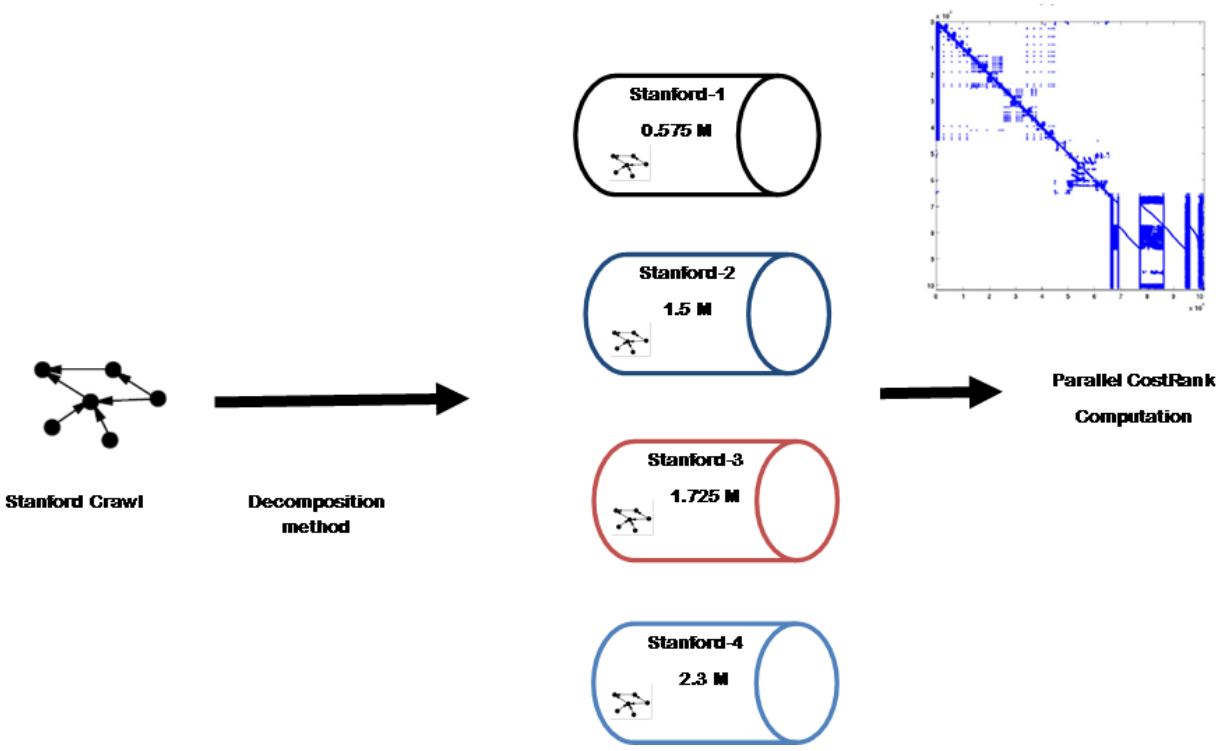


Figure D-2. The decomposition model for Stanford crawl

The bulk of the clauses and practical assignment of attack graph ranking used the value d= 0.85, but other values could be practiced according to specific environments.

The damping factor is representing the chance of an attacker continuing to fathom the current course of the attempt. Therefore, in the context of network:

$$\Re^{(t+1)}(x) = ((1-d) - \frac{n}{\sum od_i} * \frac{1}{N}) + \sum_{1}^{i} (\Re_i^{(t)} * P_R)$$ for Initial attack state

$$\Re^{(t+1)}(x) = (d - \frac{n}{\sum od_i} * \frac{1}{N}) + \sum_{1}^{i} (\Re_i^{(t)} * P_R)$$ otherwise

Where n is the number of links point to state x and N is the total number of states in a network representation graph.

A CostRank assigned to the N sates of the graph, depends on the stochastic random walker matrix multiply in the ranking vector, for each step of attack, the attacker move with probability P to the next step of the multi - step attack, if a dangling state (without out links) reached the attacker needs to move randomly to another state with probability 1. The CostRank of a specific state, represent the amount of time and efforts that the attackers spend to exploit the vulnerability to achieve that country.

The CostRank represent a stochastic course of actions of attack graph states, this process depends on the graph, the cost vectors and the damping factor.

The damping factor can take any value in the range of [> 0, <1] and it is important to monitor the ranking induced by the mutation of the damping element.

1- The experiment to calculate CostRank using a Stanford-1 data link with damping factor=0. 85.

d=. 85		
Order	CostRank	Page
1	0.028879	http://aa.stanford.edu
2	0.018288	http://aa.stanford.edu/aeroastro/AAfolks.html
3	0.01761	http://aa.stanford.edu/aeroastro/AssistantsAero.html
4	0.017065	http://aa.stanford.edu/aeroastro/EngineerAero.html
5	0.017027	http://aa.stanford.edu/aeroastro/Faculty.html
6	0.016165	http://aa.stanford.edu/aeroastro/FellowsAero.html
7	0.015583	http://aa.stanford.edu/aeroastro/GraduateGuide.html
8	0.014989	http://aa.stanford.edu/aeroastro/Labs.html
9	0.014572	http://aa.stanford.edu/aeroastro/Links.html
10	0.013452	http://aa.stanford.edu/aeroastro/MSAero.html
11	0.013385	http://aa.stanford.edu/aeroastro/News.html
12	0.012778	http://aa.stanford.edu/aeroastro/PhdAero.html
13	0.012742	http://aa.stanford.edu/aeroastro/aacourseinfo.html
14	0.012526	http://aa.stanford.edu/aeroastro/aafaculty.html
15	0.012398	http://aa.stanford.edu/aeroastro/aalabs.html
16	0.012337	http://aa.stanford.edu/aeroastro/admitinfo.html
17	0.012334	http://aa.stanford.edu/aeroastro/courseinfo.html
18	0.012318	http://aa.stanford.edu/aeroastro/draftcourses.html
19	0.012229	http://aa.stanford.edu/aeroastro/labs.html
20	0.012092	http://aa.stanford.edu/aeroastro/prospective.html
21	0.011911	http://aa.stanford.edu/aeroastro/resources.html
22	0.011873	http://aa.stanford.edu/aeroastro/visitday.html

2- The experiment to calculate CostRank using Stanford-1 data link with damping factor=0.95.

d=.95		
Order	CostRank	Page
1	0.018151	http://aa.stanford.edu
2	0.010391	http://aa.stanford.edu/aeroastro/AAfolks.html
3	0.009599	http://aa.stanford.edu/aeroastro/AssistantsAero.html
4	0.009195	http://aa.stanford.edu/aeroastro/EngineerAero.html
5	0.008988	http://aa.stanford.edu/aeroastro/Faculty.html
6	0.007925	http://aa.stanford.edu/aeroastro/FellowsAero.html
7	0.007743	http://aa.stanford.edu/aeroastro/GraduateGuide.html
8	0.007021	http://aa.stanford.edu/aeroastro/Labs.html
9	0.006595	http://aa.stanford.edu/aeroastro/Links.html
10	0.006053	http://aa.stanford.edu/aeroastro/MSAero.html
11	0.006024	http://aa.stanford.edu/aeroastro/News.html
12	0.005844	http://aa.stanford.edu/aeroastro/PhdAero.html
13	0.005838	http://aa.stanford.edu/aeroastro/aacourseinfo.html
14	0.005458	http://aa.stanford.edu/aeroastro/aafaculty.html
15	0.005221	http://aa.stanford.edu/aeroastro/aalabs.html
16	0.00496	http://aa.stanford.edu/aeroastro/admitinfo.html
17	0.004952	http://aa.stanford.edu/aeroastro/courseinfo.html
18	0.004554	http://aa.stanford.edu/aeroastro/draftcourses.html
19	0.004521	http://aa.stanford.edu/aeroastro/labs.html
20	0.004415	http://aa.stanford.edu/aeroastro/prospective.html
21	0.004194	http://aa.stanford.edu/aeroastro/resources.html
22	0.004082	http://aa.stanford.edu/aeroastro/visitday.html

3- The experiment to calculate CostRank using Stanford-1 data link with damping factor=0.99.

d=.99		
Order	CostRank	Page
1	0.013041	http://aa.stanford.edu
2	0.011202	http://aa.stanford.edu/aeroastro/AAfolks.html
3	0.009913	http://aa.stanford.edu/aeroastro/AssistantsAero.html
4	0.009824	http://aa.stanford.edu/aeroastro/EngineerAero.html
5	0.009607	http://aa.stanford.edu/aeroastro/Faculty.html
6	0.009419	http://aa.stanford.edu/aeroastro/FellowsAero.html
7	0.008052	http://aa.stanford.edu/aeroastro/GraduateGuide.html
8	0.007772	http://aa.stanford.edu/aeroastro/Labs.html
9	0.007169	http://aa.stanford.edu/aeroastro/Links.html
10	0.007082	http://aa.stanford.edu/aeroastro/MSAero.html
11	0.006958	http://aa.stanford.edu/aeroastro/News.html
12	0.006783	http://aa.stanford.edu/aeroastro/PhdAero.html
13	0.006622	http://aa.stanford.edu/aeroastro/aacourseinfo.html
14	0.006151	http://aa.stanford.edu/aeroastro/aafaculty.html
15	0.006058	http://aa.stanford.edu/aeroastro/aalabs.html
16	0.006047	http://aa.stanford.edu/aeroastro/admitinfo.html
17	0.00592	http://aa.stanford.edu/aeroastro/courseinfo.html
18	0.005915	http://aa.stanford.edu/aeroastro/draftcourses.html
19	0.005837	http://aa.stanford.edu/aeroastro/labs.html
20	0.005582	http://aa.stanford.edu/aeroastro/prospective.html
21	0.005394	http://aa.stanford.edu/aeroastro/resources.html
22	0.005354	http://aa.stanford.edu/aeroastro/visitday.html

In the experiments, using Stanford-1 link data to compute the CostRank using 0.85, 0.95, 0.99 damping factors, indicates clearly the ranking values changed significantly (not stable) according to the variations of damping factor, please refer to Figure D-3.

The result indicates as well that the damping factor (0.85) is ultimate at least for using a cost-centric AG ranking, as the convergence of CostRank algorithm iterations increased rapidly, when d=0.85 is used.

The apprehension of how the ranking value changes according to modified damping factors needs to be undertaken to get stable ranking for different applications.

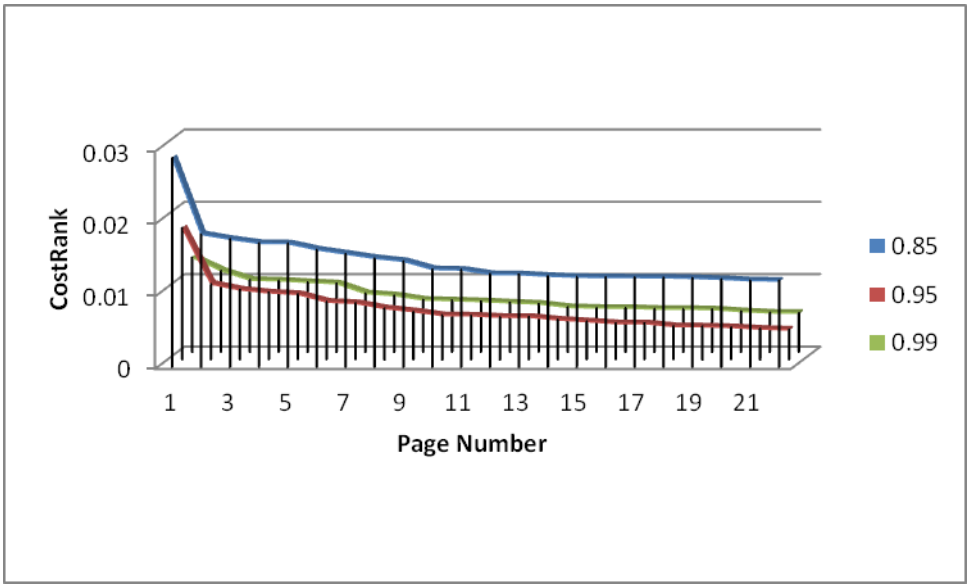
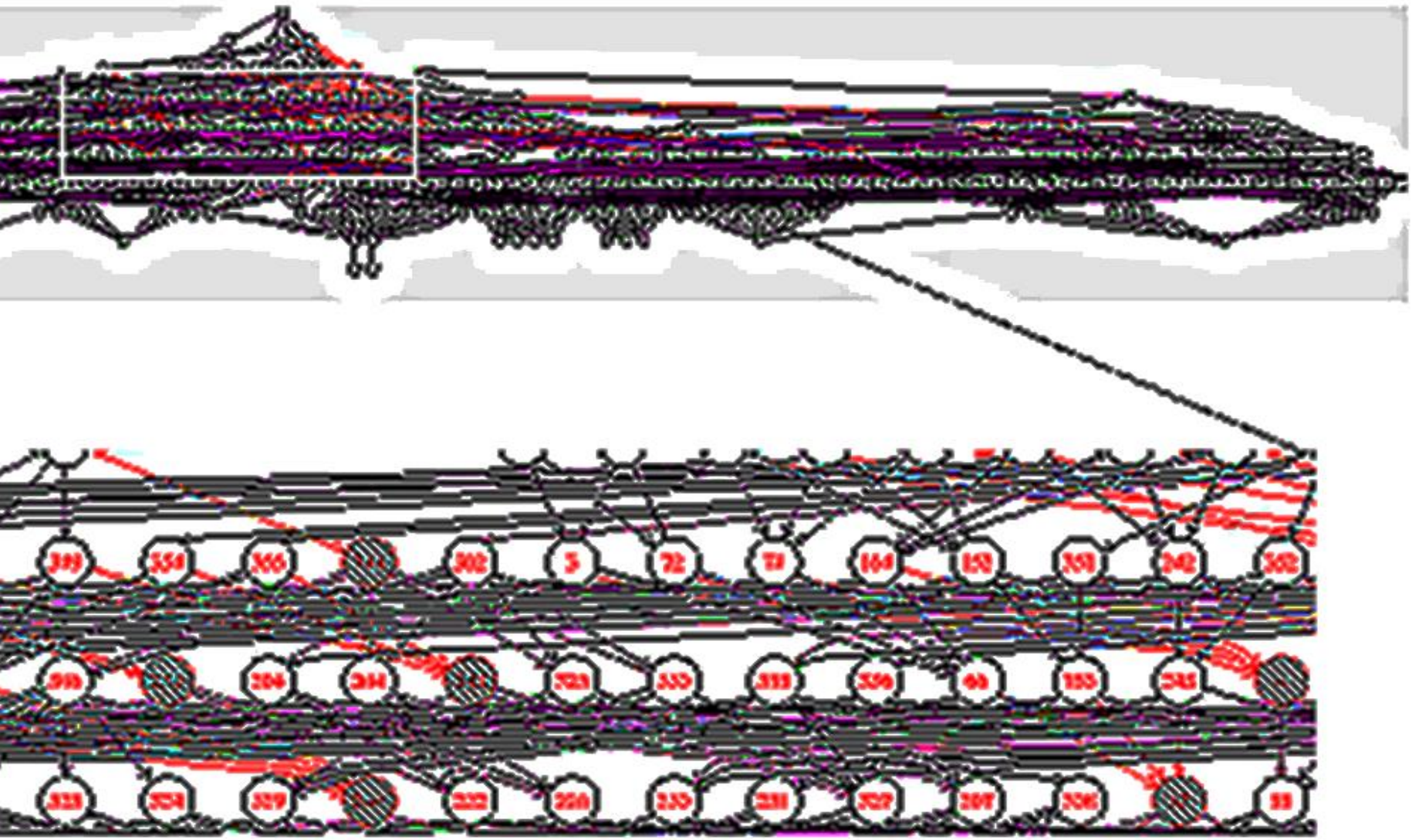


Figure D-3. CostRank calculation metric as d varies.

APPENDIX E. ATTACK GRAPH COMPLEXITY [Idika, 2010]



APPENDIX F.TABLE OF ITERATES FOR COSTRANK COMPUTATION

Iteration	CostRank											
1	[0.0375	0.0442	0.0438	0.0352	0.0463	0.0608	0.0768	0.0584	0.0719	0.1001	0.1337	0.1413]
2	[0.0294	0.0215	0.0252	0.0200	0.0488	0.0635	0.0607	0.0583	0.0546	0.1001	0.1740	0.2456]
3	[0.0478	0.0169	0.0123	0.0115	0.0488	0.0635	0.0661	0.0461	0.0545	0.1001	0.1740	0.3197]
4	[0.0188	0.0275	0.0096	0.0056	0.0488	0.0635	0.0661	0.0502	0.0431	0.1001	0.1740	0.3197]
5	[0.0434	0.0513	0.0501	0.0386	0.0523	0.0701	0.0890	0.0673	0.0825	0.1150	0.1559	0.1642]
6	[0.0435	0.0514	0.0503	0.0390	0.0526	0.0703	0.0892	0.0675	0.0828	0.1154	0.1561	0.1645]
7	[0.0436	0.0515	0.0505	0.0394	0.0529	0.0705	0.0894	0.0677	0.0831	0.1158	0.1563	0.1648]
8	[0.0437	0.0516	0.0507	0.0398	0.0532	0.0707	0.0896	0.0679	0.0834	0.1162	0.1565	0.1651]
9	[0.0438	0.0517	0.0509	0.0402	0.0535	0.0709	0.0898	0.0681	0.0837	0.1166	0.1567	0.1654]
10	[0.0439	0.0518	0.0511	0.0406	0.0538	0.0711	0.0900	0.0683	0.0840	0.1170	0.1569	0.1657]
11	[0.0390	0.0519	0.0513	0.0410	0.0541	0.0713	0.0902	0.0685	0.0843	0.1174	0.1571	0.1660]
12	[0.039	0.0519	0.0513	0.041	0.0541	0.0713	0.0902	0.0685	0.0843	0.1174	0.1571	0.166]
13	[0.0441	0.0520	0.0515	0.0414	0.0544	0.0715	0.0904	0.0687	0.0846	0.1178	0.1573	0.1663]
14	[0.0441	0.0520	0.0515	0.0414	0.0544	0.0715	0.0904	0.0687	0.0846	0.1178	0.1573	0.1663]
15	[0.0441	0.0520	0.0515	0.0414	0.0544	0.0715	0.0904	0.0687	0.0846	0.1178	0.1573	0.1663]
16	[0.0441	0.0520	0.0515	0.0414	0.0544	0.0715	0.0904	0.0687	0.0846	0.1178	0.1573	0.1663]
17	[0.0441	0.0520	0.0515	0.0414	0.0544	0.0715	0.0904	0.0687	0.0846	0.1178	0.1573	0.1663]
18	[0.0441	0.0520	0.0515	0.0414	0.0544	0.0715	0.0904	0.0687	0.0846	0.1178	0.1573	0.1663]
19	[0.0441	0.0520	0.0515	0.0414	0.0544	0.0715	0.0904	0.0687	0.0846	0.1178	0.1573	0.1663]
20	[0.0441	0.0520	0.0515	0.0414	0.0544	0.0715	0.0904	0.0687	0.0846	0.1178	0.1573	0.1663]
21	[0.0441	0.0520	0.0515	0.0414	0.0544	0.0715	0.0904	0.0687	0.0846	0.1178	0.1573	0.1663]
22	[0.0441	0.0520	0.0515	0.0414	0.0544	0.0715	0.0904	0.0687	0.0846	0.1178	0.1573	0.1663]
23	[0.0441	0.0520	0.0515	0.0414	0.0544	0.0715	0.0904	0.0687	0.0846	0.1178	0.1573	0.1663]
24	[0.0441	0.0520	0.0515	0.0414	0.0544	0.0715	0.0904	0.0687	0.0846	0.1178	0.1573	0.1663]